UNIVERSIDAD CARLOS III DE MADRID
GRADO EN INGENIERÍA INFORMÁTICA

Proyecto de Fin de Grado

Desarrollo de aplicación para el diseño de escenarios de juegos educativos

Autor: Miguel Carvajal Jesús

Tutor: Telmo Zarraonandia Ayo

Junio de 2012

Agradecimientos

En primer lugar, a mis padres por todo el apoyo que me han dado siempre y por estar ahí en todo momento, confiando en mí y ayudándome a seguir. Por su infinita paciencia y comprensión. Sin ellos esto no hubiera sido posible.

A mis amigos, por saber actuar en cada instante y compartir mis alegrías.

A mi familia, y a los compañeros de carrera y profesores que han realizado este camino conmigo.

A Telmo Zarraonandia, por sus horas de dedicación como tutor de este proyecto y su actitud predispuesta en todo momento.

TABLA DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

La finalidad de este primer capítulo es proporcionar al lector una visión general de este proyecto, explicando su contexto, motivación, los objetivos planteados y la forma en que se espera que sean satisfechos. De igual forma se explica la estructura del resto del documento.

1.1 CONTEXTO

Este proyecto se enmarca dentro del área de la educación asistida por ordenador, rama de la informática que tiene por objeto tratar de aprovechar las ventajas que ofrece la tecnología para mejorar los procesos educativos. Entre estas ventajas destaca la capacidad de transmitir contenidos y valores presentados de una forma atractiva a los menores, mediante los cuales se pueden adquirir habilidades relacionadas con este contexto digital, como es la destreza o la coordinación.

Dentro de esta área en los últimos años está cobrando cada vez mayor importancia el denominado *game based learning* o aprendizaje soportado mediante videojuegos. Esto consiste en la aplicación de las nuevas tecnologías como base para mejorar el aprendizaje y rendimiento de niños que tan fuertemente atraídos se sienten por la era digital. Este aprendizaje basado en el juego persigue profundizar en el aprendizaje a través de experiencias lúdicas y a la vez motivadoras. [1]

Los motivos por los que este método de aprendizaje está cobrando popularidad son que los educadores están descubriendo la eficacia de los videojuegos con la finalidad de aumentar la motivación y el interés del estudiante en la materia en cuestión [2,3], y esto ha sido demostrado en numerosas áreas educativas, como la medicina, las ciencias, el entrenamiento militar o la formación en situaciones de emergencia. [4]

Sin embargo, el diseño y desarrollo de videojuegos no es para nada sencillo. La combinación entre la educación y el entretenimiento mediante el uso de estas tecnologías supone altos costes. Es por ello que surge la necesidad de herramientas o modelos que faciliten el desarrollo y la personalización de los juegos para procesos formales de aprendizaje. La creación de métodos y aplicaciones de autoría para el desarrollo de los juegos pueden fomentar a la reutilización de componentes de juegos, que subsanen en gran medida los altos costos que supone el desarrollo en esta área.

1.2 MOTIVACIÓN

El autor del artículo *Seeking Reusability of Computer Games Designs for Informal Learning* [5], propone un modelo de diseño de videojuegos educativos que considera distintas perspectivas y componentes que pueden ser tratadas de forma independiente y luego ensambladas y combinadas entre si, de tal forma que se simplifiquen las tareas de diseño y se facilite la reutilización de las distintas partes, reduciendo de esa manera los costes asociados a dicha tarea.

Una de esas perspectivas es el escenario, que proporciona los medios para representar los elementos que compondrán el juego en sí. La definición del escenario queda determinada por un conjunto de escenas que están interconectadas entre sí. Estas conexiones surgen a través de enlaces y cada escena representa una situación o ambiente diferente. El diseñador del escenario puede definir una entidad de la escena por cada área de la escena que ofrece la posibilidad de interactuar con estos elementos, representando visualmente o por medio de sonidos el estado de cada objeto.

La realización de este proyecto surge de la necesidad de proveer un software que permita la descripción de la composición de escenas de videojuegos educativos de forma sencilla y adecuada para el personal docente, sin tener necesariamente un perfil técnico. De esta forma debe ofrecer un proceso metódico y sencillo para la creación de los escenarios definidos anteriormente, siendo en todo momento intuitivo y con capacidad para generar descripciones que puedan ser luego entendidas y procesables por otra aplicación, encargada de generar el escenario virtual donde tendrá lugar el juego.

La aplicación desarrollada permite al educador o educadores que hacen uso de ella realizar, de una manera intuitiva y sencilla, escenarios para dicho videojuego. De esta forma el instructor, mediante la herramienta, tiene total capacidad para desarrollar el escenario en el que se desarrollará la acción, siendo el mismo el principal conductor de las ideas que desea implementar. De esta forma puede plasmar en el escenario los elementos de tal forma que le permita recrear situaciones con el fin de satisfacer diversos aspectos educativos orientados al alumno.

1.3 OBJETIVOS DEL PROYECTO

1.3.1 OBJETIVO PRINCIPAL

El objetivo principal del proyecto es proporcionar una herramienta de autoría para la creación de escenarios de videojuegos, capaz de generar un documento XML que contenga toda la información necesaria para que, mediante el uso de otra aplicación externa y el motor de desarrollo Unity 3D, se pueda generar el escenario del videojuego.

1.3.2 OBJETIVOS SECUNDARIOS

Además, la aplicación debe presentar una interfaz sencilla, y el uso de la misma ha de ser intuitivo, con el fin de que el proceso de creación pueda ser completado correctamente de la manera más sencilla posible sin requerir del usuario un perfil técnico o una extensa formación en la herramienta.

1.4 ESTRUCTURA DEL DOCUMENTO

El presente documento se encuentra divido en varios capítulos, recogiendo toda la información relacionada con el desarrollo de este proyecto. A continuación se ofrece un resumen del contenido de cada uno de ellos.

CAPÍTULO 1 – INTRODUCCIÓN.

En este capítulo se establece el contexto que abarca el desarrollo del proyecto, describiendo brevemente las ideas del mismo y los objetivos que se desean alcanzar, presentando motivaciones y soluciones para satisfacer las necesidades del problema planteado.

CAPÍTULO 2 – ESTADO DEL ARTE.

Durante este capítulo se ofrece una visión global acerca de lo relacionado con los videojuegos y la educación, analizando la situación actual referente a dicho tema. Además, se realiza un estudio de las tecnologías disponibles que se pueden utilizar para llevar a cabo la realización del proyecto, con el fin de obtener las más adecuadas al trabajo.

CAPÍTULO 3 – ANÁLISIS DEL SISTEMA.

Aquí se recoge toda la información relativa a la fase de análisis del sistema. Quedan definidos tanto casos de uso como requisitos de software y todo ello sirve de base para la siguiente etapa del proyecto.

CAPÍTULO 4 – DISEÑO DEL SISTEMA.

Este capítulo está dedicado a la fase de diseño del proyecto. Basándose en las necesidades del sistema y la información de la etapa de análisis, se crean modelos de arquitectura de alto nivel que detallen los componentes que forman el sistema, con el fin de llevar a cabo su desarrollo. También se ofrece un prototipo de la interfaz del sistema, que facilita la comprensión del sistema.

CAPÍTULO 5 – IMPLEMENTACIÓN.

En este apartado se recogen los aspectos más importantes del desarrollo del proyecto, incluyendo tecnologías utilizadas y fragmentos de código relevantes para su implementación.

CAPÍTULO 6 – GESTIÓN DEL PROYECTO.

Este capítulo trata de cómo ha sido gestionado el proyecto, las metodologías utilizadas para llevarlo a cabo, tales como el ciclo de vida seguido. Además se ofrece la planificación inicial y final, realizando el cálculo del presupuesto.

CAPÍTULO 7 – CONCLUSIONES Y LÍNEAS FUTURAS.

Este apartado recoge las conclusiones extraídas a lo largo del desarrollo del proyecto, basándose en los objetivos iniciales y el proceso completo de realización. A su vez se describen los conocimientos aprendidos durante el desarrollo y algunos aspectos a tener en cuenta en ocasiones futuras.

CAPÍTULO 8 – REFERENCIAS Y BIBLIOGRAFÍA.

Referencias a libros, documentos, webs y conceptos aparecidos a lo largo del documento.

2. ESTADO DEL ARTE

INTRODUCCIÓN

El objetivo de este capítulo es presentar aquellas tecnologías y trabajos relacionados con el presente proyecto. La finalidad es facilitar al lector la comprensión de los siguientes capítulos y esclarecer todos los conceptos de forma que pueda valorar adecuadamente las contribuciones del proyecto.

El producto está basado en la autoría de creación de escenarios para videojuegos educativos, es por ello que a lo largo de este capítulo se hablará de la educación por medio de los videojuegos, y las ventajas que supone la utilización de estas herramientas con el fin de educar y enseñar.

Además, se aborda el estudio y evaluación de diversas tecnologías existentes, su estado actual, y el porqué de su utilización en el desarrollo de esta aplicación.

2.1 VIDEOJUEGOS EN LA EDUCACIÓN

2.1.1 BREVE HISTORIA DE LOS VIDEOJUEGOS

Se puede situar la aparición del primer videojuego en la década de los años 40, durante la cual, técnicos norteamericanos desarrollador el primer simulador de vuelo con la finalidad del entrenamiento de los pilotos.

A partir de los años 60, con la aparición del primer microprocesador y la tercera generación de ordenadores, el desarrollo de videojuegos con fines meramente lúdicos no ha parado de crecer.

Así, en el año 1972, aparece 'PONG' [Ilustración 1], el primer videojuego de uso doméstico. Su funcionamiento consistía en una básica simulación de una partida de tenis de mesa. En los años siguientes, la marca *Atari* es la encargada de comercializarlo, lanzando al mercado el primer sistema de videojuegos basado en cartuchos, alcanzando un gran éxito en Estados Unidos y provocando simultáneamente las primeras preocupaciones sobre los posibles efectos negativos de los videojuegos en la conducta de los menores. [6]

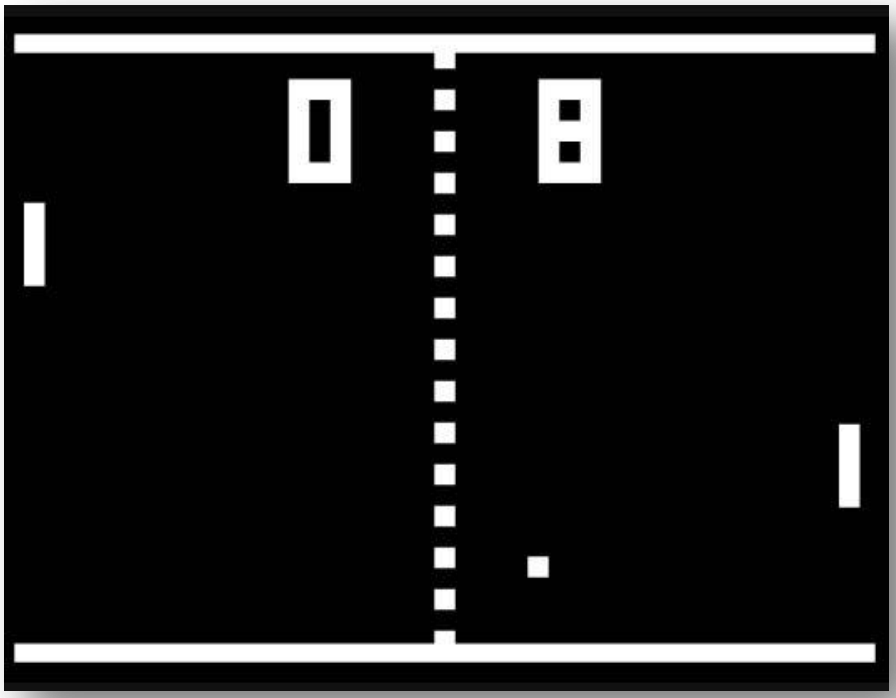


Ilustración 1. Videojuego PONG lanzado en 1972

Los videojuegos sufrieron una gran evolución a lo largo de la década de los 80, destacando el lanzamiento de nuevos sistemas de videojuegos por parte de dos populares compañías del sector: Sega y Nintendo, acompañados de otros sistemas que supusieron una revolución como son los ordenadores domésticos de 8 bits, *Spectrum* y *Amstrand*, haciendo que estos sistemas se convirtieran en poco tiempo en los juguetes más demandados por los niños. [7]

En la actualidad los videojuegos pueden ser encontrados en múltiples plataformas, desde los sistemas de videojuegos de última generación hasta dispositivos móviles, pasando por ordenadores y videoconsolas portátiles, entre otros.

A continuación se realizará un estudio acerca de la relación entre los videojuegos y la educación.

2.1.2 LOS VIDEOJUEGOS Y LA EDUCACIÓN

Los videojuegos han despertado, desde el inicio de su desarrollo, gran interés de investigación sobre su influencia en la conducta del ser humano.

Las primeras investigaciones sobre estos tipos de juegos de carácter electrónico sucedieron de forma simultánea a su aparición, a comienzos de la década de los setenta, como se ha comentado en el apartado anterior. Los primeros estudios se centraban en el análisis de sus posibles efectos psicológicos. No es hasta finales de esta década y principios de los años ochenta, cuando estos estudios van más allá y

comienzan a plantear el potencial educativo que pueden desarrollar los videojuegos sobre niños y adolescentes. [8]

Uno de los investigadores pioneros en este campo es G. H. Ball, con su estudio titulado '*Telegames teach more than you think*' [9] donde se analiza por primera vez el potencial de este tipo de juegos para desarrollar capacidades espaciales, profundizando en los aspectos tridimensionales y de simulación de formas reales. A su vez, se obtuvieron diversas conclusiones tales como que la utilización de estos juegos electrónicos favorece la destreza intelectual de los menores, facilitando la comprensión lectora, la asimilación de conceptos numéricos e, incluso, estimulando la lectura.

Además, un experimento realizado por J. Griffith en 1983 [10] demostró que el uso de los videojuegos podía potenciar la destreza en la coordinación tanto visual como motora, al igual que la capacidad de coordinación óculo-manual, cualidad potencialmente desarrollada en personas asiduas a este tipo de juegos, mejorando así los reflejos y respuestas en habilidades motoras.

Uno de los aspectos más característicos de los videojuegos es la capacidad de atracción y captación de atención de los más pequeños, que supone para los mismos el suficiente atractivo y motivación. Esto es debido a algunos de las siguientes características: [11]

- Poseen un carácter entretenido, que actúa en conjunto con altos niveles de estimulación visual y auditiva.

- En ellos se implementan diversos niveles de dificultad que avanzan de manera progresiva, haciendo que se requiera un dominio de cada fase, enfrentando al usuario a un reto de superación. Esto supone la gratificación de poder solucionar un problema más o menos complejo, actuando como un logro de objetivo, fundamento base de la educación.

- Los retos que suceden de forma gradual precisan de una constante superación personal.

- Surge el concepto de competitividad, potenciando la atracción del aprendizaje en el sentido de conocer quién supera el mayor número de objetivos, y mejorando la intención de superación individual.

- Todo esto afecta al concepto de la autoestima, creciendo a medida que se superan diferentes retos.

Basándose en el conjunto de investigaciones analizadas se puede obtener como conclusión que el uso de este tipo de juegos como finalidad de determinados aprendizajes es bastante positivo en diferentes aspectos. Los videojuegos permiten incrementar la motivación para el conocimiento de diversas materias y conjuntos de enseñanza. [12]

Incluso el uso de los videojuegos puede favorecer también en el aspecto de terapias psicológicas y fisiológicas, al igual que facilitando diversas dificultades de aprendizaje, ayudando a la recuperación de ciertas destrezas de carácter físico o psicológico y mejorando el rendimiento.

2.1.2.1 VIDEOJUEGOS EDUCATIVOS POPULARES

En este apartado se nombraran algunos videojuegos de carácter educativo que han sido populares a lo largo de la historia.

BRAIN TRAINING DEL DR. KAWASHIMA

Este juego fue desarrollado para la plataforma Nintendo DS en el año 2005. Se trata de un videojuego de lógica y puzles, que fomenta la capacidad cerebral de sus usuarios. Su estructura se basa en una serie de ejercicios cuya finalidad es ejercitar la mente. Las actividades que engloba son de diverso carácter, desde ejercicios de cálculo, retención de datos, ejercicios de lectura, ortografía hasta juegos de agilidad mental, como sudokus.

Ilustración 2. Nivel matemático de Brain Training

El juego presenta unos gráficas donde el usuario puede consultar los datos de sus resultados en cada una de las actividades, pudiendo apreciar su evolución en cada

ejercicio y de forma general, permitiéndole analizar su progreso y los puntos débiles que posee, con el objetivo de mejorar en dichos aspectos.

Este videojuego [13] no está exclusivamente realizado para niños, incluso su intención es también la de dirigirse a un público más adulto, definiéndose como un potenciador de las capacidades mentales y rejuvenecedor de la edad cerebral, presentándose como un producto bastante beneficioso para personas de la tercera edad.

APRENDE CON PIPO

Esta saga de videojuegos esta orientada en su totalidad a los menores. Su finalidad es enseñar y educar a los niños, definiéndose como software educativo para menores. En los videojuegos de esta saga se emplean actividades diarias, como pueden ser los deportes y la visita a lugares comunes como el parque y la escuela, con el fin de estimular a los más pequeños.

Ilustración 3. Nivel de 'Aprende con Pipo'

Además posee diferentes juegos divididos en niveles dependiendo de la edad del menor, posee una saga orientada a niños de hasta los tres años, con juegos simples de carácter intuitivo, donde las enseñanzas que presentan son básicas. Otra de las sagas está orientada a niños de entre 6 y 8 años, donde el nivel de enseñanza es más elevado, pero continúa con su aspecto sencillo e intuitivo. Incluso existe un tercer ciclo de videojuegos de esta saga desarrollado para niños de hasta 12 años. En este caso los

juegos de ‘Aprende con Pipo’ pueden presentarse como un buen complemento a la educación del menor.

ENGLISH TRAINING

Este videojuego [14] pretende que el usuario desarrolle un área de conocimiento basada en el lenguaje, en este caso el inglés (pero existen otros juegos de este mismo tipo en los que se profundiza en otros idiomas, un ejemplo de ello es ‘Mi experto en Francés’).

El objetivo no es aprender este idioma desde cero, sino reforzar los conceptos que el usuario posee, ampliando a su vez vocabulario. Este juego está enfocado a gente que no practica frecuentemente el idioma y debe acostumbrarse a oír hablar en él.

Ilustración 4. Juego educativo English Training

El juego está estructurado en varias lecciones distribuidas por tipos y niveles, y estos están acompañados de diferentes minijuegos que ayudan a refrescar lo recién aprendido de una forma divertida y mejorando la comprensión oral.

2.2 HERRAMIENTAS DE DESARROLLO

2.2.1 MOTORES DE VIDEOJUEGOS

La aplicación a desarrollar, como se ha explicado en secciones anteriores, debe actuar como autoría de creación de un escenario de videojuegos educativos que será más tarde generado en tres dimensiones por medio de otra herramienta externa, en este caso, un motor de videojuego.

Existen en el mercado varias herramientas de desarrollo de videojuegos, un ejemplo de ellas son Torque y Unity 3D, descritas a continuación.

2.2.1.1 TORQUE

Se trata de un motor de desarrollo de videojuegos [15] del que se necesita la licencia para hacer uso de él. Una vez adquirida se ofrece el código fuente en C++ que se puede modificar a conveniencia con el fin de optimizar la herramienta o alcanzar otros niveles de desarrollo según sea necesario.

Es un producto multiplataforma, y se programa mediante misiones en terrenos extensos. Posee un potente creador de terrenos, en el que se puede añadir un gran número de elementos.

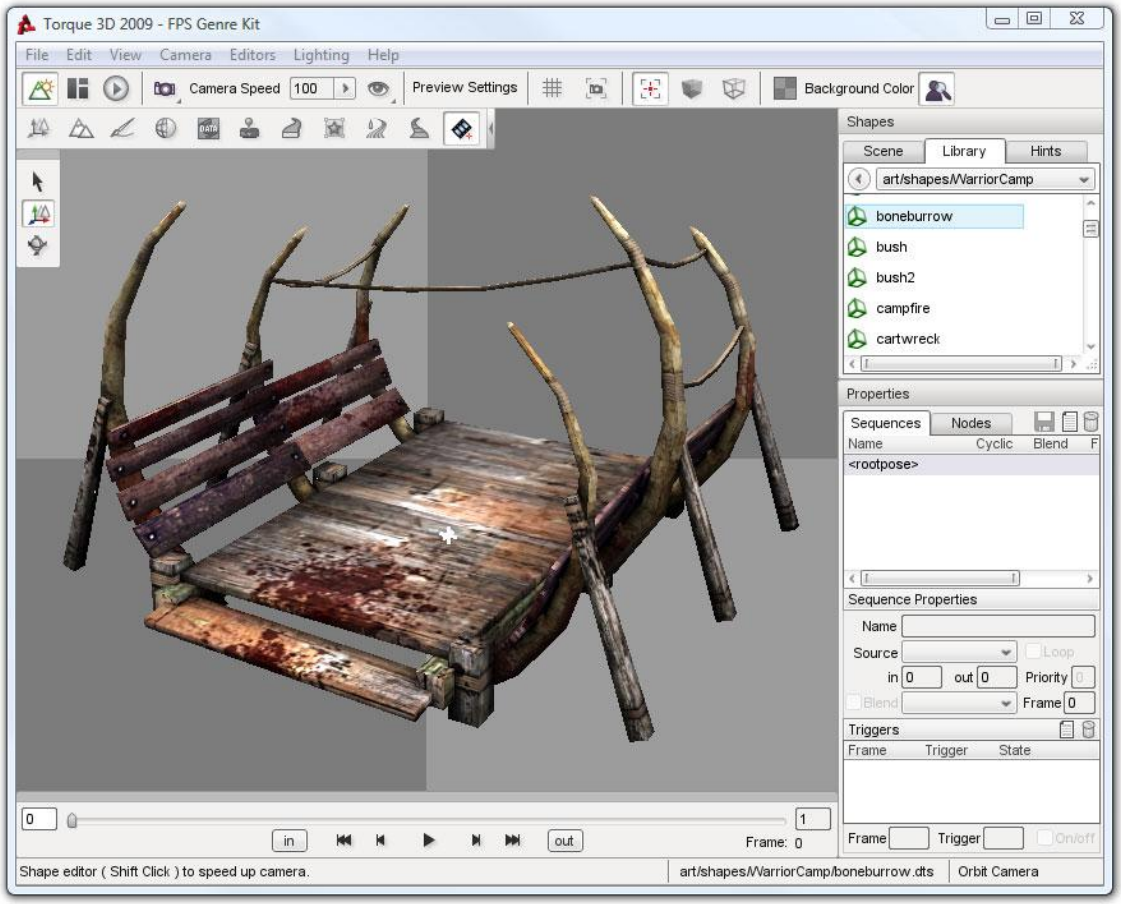


Ilustración 5. Editor de Torque 3D

Otra de sus características es que todos los juegos desarrollados por esta herramienta presentan una arquitectura Cliente-Servidor, dando opción a que el juego pueda ser ejecutado sin problemas con un gran número de jugadores online.

Además posee una curva de aprendizaje muy corta, pudiendo desarrollar en poco tiempo juegos con gran potencial.

2.2.1.2 UNITY 3D

Este motor tridimensional de desarrollo de videojuegos está desarrollado por la empresa *Unity Technologies* y es denominado como *Unity 3D*. [16]

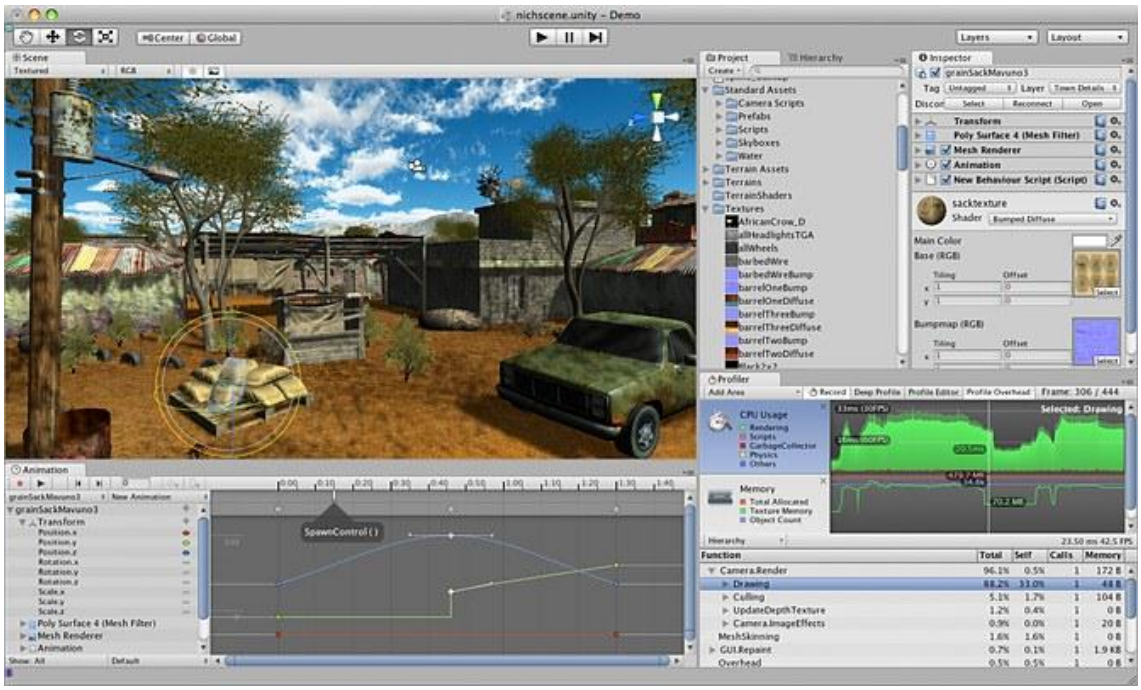


Ilustración 6. Editor de Unity 3D

Se trata de un motor que permite crear juegos para multitud de plataformas, entre ellas: *Windows, Mac, Xbox 360, PlayStation 3, Wii, iPad y Android*. Y además, gracias a una extensión web desarrollada por *Unity Technologies*, también permite el desarrollo de juegos de navegador, tanto para *Windows* como para *Mac*.

Unity 3D es uno de los motores más versátiles del momento, y ofrece multitud de opciones a aquellos que desean iniciarse en el desarrollo de juegos multiplataforma. Además existe mucha documentación sobre este motor y número ayuda en múltiples formatos para comenzar a desarrollar utilizándolo.

Se basa en la creación de scripts, que definen el comportamiento del videojuego creado con esta herramienta. Estos pueden ser creados tanto en *Javascript*, como en *C#* y *Boo* en *Unity 3D*. Esto se complementa al diseño de la superficie o terreno, sobre el que actuarán los elementos según los scripts concretados anteriormente. [17]

2.3 SITUACIÓN ACTUAL DEL DESARROLLO DE APLICACIONES

La elaboración del proyecto consiste en el desarrollo de una aplicación web que permitirá generar documentos XML que describen escenarios de videojuegos que serán interpretados por un motor de videojuegos, desarrollado mediante la herramienta Unity. Por ello, se describirán tecnologías relacionadas con el desarrollo de este tipo de aplicaciones de carácter web.

2.3.1 SITUACIÓN ACTUAL DE APLICACIONES WEB

Nos encontramos en una situación de pleno auge del desarrollo de software, en el que cada vez son más las empresas que escogen la opción de desarrollar aplicaciones con el fin de obtener beneficios a través de ellas, ya sean grandes o pequeñas. Además, no sólo son utilizadas para gestionar de forma interna la empresa en sí, también son utilizadas para darse a conocer y obtener beneficios a través de ellas.

Es una manera de optimizar recursos y de mentalización con el fin de adaptar tanto a empleados como a clientes a los conceptos de simplificación que surge mediante el uso de diverso tipo de software. [18]

Un ejemplo claro es el desarrollo de aplicaciones web con estos fines. Pueden servir para obtener información de clientes, generación de documentos necesarios, consultas de acciones realizadas por la empresa, pedidos, facturaciones, tareas a desarrollar, visualización de estadísticas, etc. Infinidad de posibilidades de acceso a la información desde cualquier punto, gracias a la evolución del software y la introducción de Internet.

Esto ha dado paso a una evolución en dichas aplicaciones. Durante los últimos años ha ido modificándose el proceso de desarrollo de las mismas, ahora surgen aplicaciones dinámicas, basadas en la constante actualización e incluso permitiendo un alto nivel de personalización, de manera que puedan ser adaptadas a diferentes tipos de usuario. Esto ha encontrado su medio ideal en la web, con el fin de no hacer costoso su mantenimiento y modificación.

Una de las tecnologías que extienden la funcionalidad de los servidores web, y hace posible lo descrito anteriormente, son las denominadas *Java Server Pages* o *JSPs*. Estas permiten unificar HTML, aplicaciones desarrolladas en Java y otros componentes como las *Java Beans*, dando lugar a una página web especial compilada de forma dinámica cuando es llamada.

A esta tecnología es posible añadirle otras que dinamicen aún más la aplicación, como es el caso del uso de Ajax, una técnica que permite crear módulos interactivos en la aplicación, ejecutadas directamente en el cliente, reduciendo el tiempo de respuesta para aumentar la interactividad y usabilidad en las aplicaciones. [19]

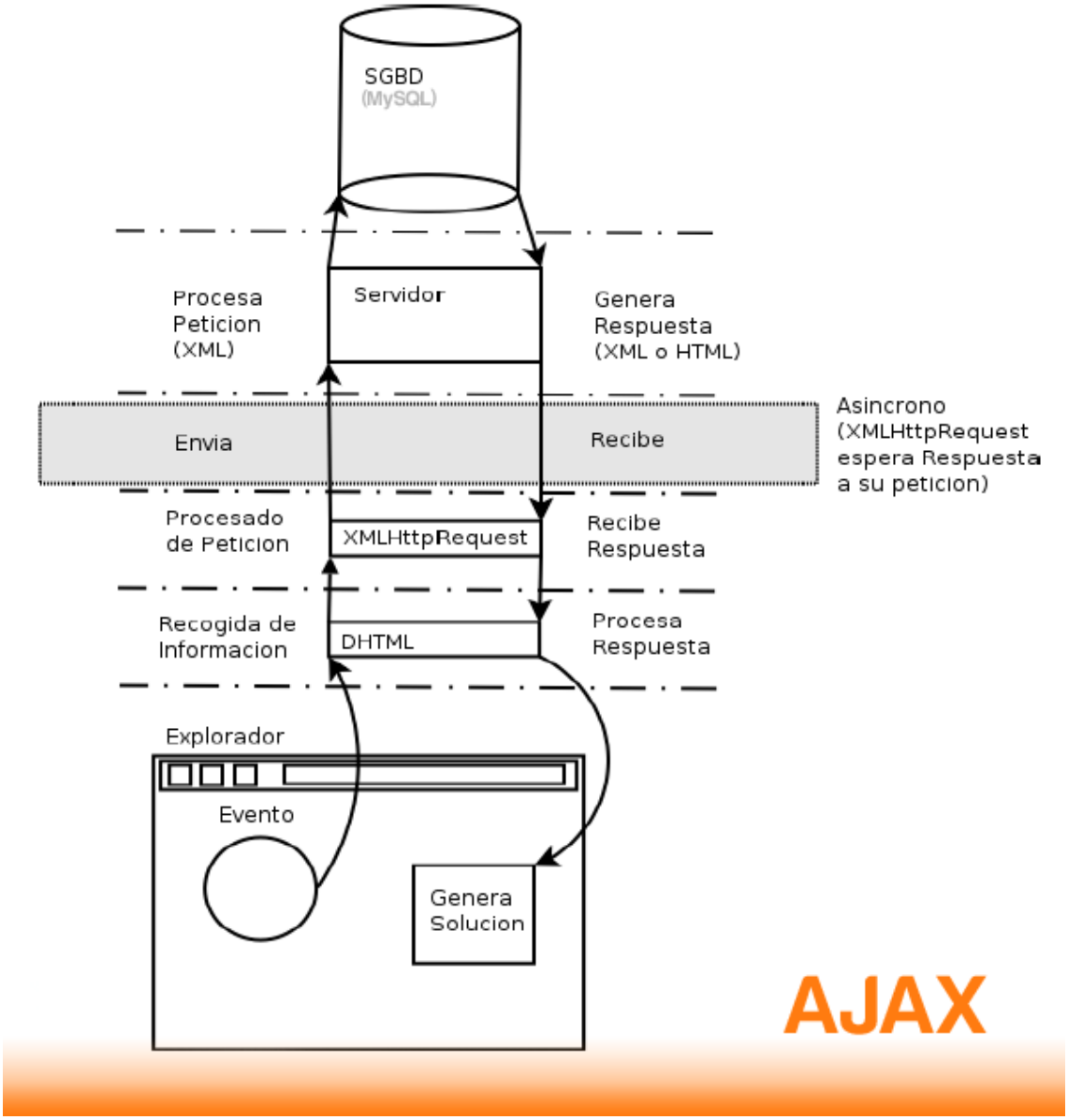


Ilustración 7. Esquema de la tecnología Ajax

La ilustración [Ilustración 7] anterior muestra un esquema de la tecnología descrita anteriormente.

2.4 TECNOLOGÍAS

Es necesario conocer todas y cada una de las características de las posibles tecnologías a utilizar con el fin de conocer cual es la que se adecua de manera más óptima al proyecto, cubriendo las necesidades del mismo. Por esta razón en los siguientes apartados se describirá el conjunto de ventajas o inconvenientes que pueden presentar unas y otras, analizándolo desde el punto de vista de adecuación al proyecto.

2.4.1 APLICACIONES DE ESCRITORIO VS. APLICACIONES WEB.

La pregunta inicial a responder es la elección entre una aplicación web o una aplicación de escritorio.

Para ello se debe recuperar toda la información posible ante ambas opciones y realizar un estudio de sus ventajas y desventajas basándose en los conceptos del proyecto.

2.4.1.1 APLICACIONES WEB

En el caso de las aplicaciones web, las características que hacen favorable la elección de esta opción son algunas de las siguientes:

+ Se tratan de aplicaciones ligeras, por lo que no es necesario contar con un equipo con excesivas prestaciones para trabajar con ellas.

+ La portabilidad de las mismas casi carecen de límites, pueden ser ejecutadas desde cualquier ordenador con acceso a internet.

+ Su actualización y mantenimiento son muy sencillos.

+ Es totalmente independiente del sistema operativo instalado en el equipo del usuario, evitando así la mayoría de problemas de compatibilidad.

+ Su distribución e instalación están caracterizadas por su comodidad, evitando restricciones y limitaciones en este aspecto.

Pero también se debe tener en cuenta que estas aplicaciones de carácter web también poseen ciertas limitaciones:

- Es necesario disponer de una conexión a Internet para la ejecución de la aplicación. Además debe tratarse de una conexión que no se encuentre demasiado limitada, puesto que existe una comunicación constante entre la aplicación y el servidor.

- El servidor debe presentar las prestaciones necesarias para que la ejecución sea lo más fluida posible, además se debe tener en cuenta la conexión simultánea entre varios usuarios.

- Es más dificultoso el desarrollo al hacerlo compatible con diversos navegadores.

- El tiempo de respuesta no es más óptimo que el de las aplicaciones de Escritorio, además depende en gran medida de la conexión del usuario.

Ilustración 8. Proceso de ejecución de aplicación web

En la ilustración anterior se puede observar el proceso que surge con la ejecución de una aplicación web: [20]

1. El usuario ingresa la URL de la aplicación desde su navegador, y este será el encargado de realizar una solicitud al servidor web mediante el protocolo HTTP.
2. El servidor Web recibe la solicitud y ejecuta el código del servidor conectando con la base de datos para recuperar la información necesaria.
3. El servidor envía la respuesta al usuario después de procesar la página.

2.4.1.2 APLICACIONES DE ESCRITORIO

En el caso de las aplicaciones de escritorio, deben ser instaladas en el equipo del usuario y son ejecutadas directamente por el sistema operativo del mismo.

En esta ocasión, las ventajas que presenta la utilización de este tipo de aplicaciones son las siguientes:

+ La ejecución, generalmente, se realiza de forma local sin accesos al exterior. Esto hace que la velocidad de procesamiento en la funcionalidad sea mayor, y permite mayores capacidades en la programación de herramientas con mayor complejidad.

+ Presentan mayor robustez ye estabilidad que las aplicaciones web.

+ El tiempo de respuesta es más óptimo en este caso, pero también depende de más factores como la memoria RAM, espacio de disco, etc.

Pero, como en el caso anterior, también presentan ciertos inconvenientes en el uso de este tipo de aplicaciones.

- El acceso está limitado por el equipo en el que es ejecutada la aplicación.

- Son totalmente independientes del sistema operativo y de las características del equipo desde el cual se utiliza dicha aplicación.

- Tanto el sistema de instalación como el de actualización son personalizadas.

- Generalmente presentan requerimientos exclusivos de software y librerías.

Ilustración 9. Proceso de ejecución de aplicación de escritorio

La ilustración anterior representa lo que ocurre en la ejecución de una aplicación de escritorio, previamente instalada en el equipo del usuario.

1. El usuario carga la aplicación.
2. El código de la herramienta se conecta a la Base de datos recuperando la información necesaria.
3. La aplicación procesa el resultado mostrándolo al usuario.

2.4.1.3 CONCLUSIÓN

Ahora que se conocen las características de ambos tipos de aplicación, se debe tener en cuenta cada una de las necesidades de la herramienta a desarrollar. En este caso es necesario desarrollar una aplicación con una interfaz intuitiva y no demasiado compleja, que sea capaz de utilizarla por un educador o personal docente. Además, es posible que sea utilizada por varios educadores desde distintos equipos, con el fin de crear distintos escenarios desde su ubicación, por ello sería más conveniente no limitar su utilización a un único lugar de trabajo, y facilitar su acceso sin limitaciones como instalaciones individuales. Las previsiones de actualización de la herramienta son factibles, sobre todo teniendo en cuenta el mantenimiento de las texturas o entidades, que irán incrementando a medida que los educadores lo vean necesario, por ello debería tenerse en cuenta todos los inconvenientes que supondrían en este aspecto la utilización de una aplicación de escritorio. Además, es conveniente que la herramienta pueda ser utilizada desde cualquier sistema operativo, evitando así gran cantidad de limitaciones.

Teniendo en cuenta todas y cada una de las características descritas anteriormente, se decide que la herramienta sea desarrollada como aplicación web, cumpliendo así prácticamente la totalidad de las necesidades que presentaba la descripción de la herramienta en su utilización.

2.4.2 LENGUAJES DE PROGRAMACIÓN Y ENTORNOS DE DESARROLLO

En el apartado anterior, se ha descrito el porqué de la elección de aplicación web frente a la aplicación de escritorio. Es el momento de estudiar qué herramientas son las más apropiadas para el desarrollo de dicha aplicación, al mismo tiempo que se selecciona el lenguaje de programación que más conveniente se cree para su realización.

2.4.2.1 LENGUAJE JAVA Y ENTORNO DE PROGRAMACIÓN ECLIPSE

JAVA

Java es un lenguaje de programación orientado a objetos, que toma gran parte de su sintaxis de los lenguajes *C* y *C++*, pero que posee un modelo de objetos con mayor simplicidad y eliminando conceptos como la manipulación directa de punteros o memoria, reduciendo así algunos errores frecuentes inducidos en los otros lenguajes nombrados. La memoria se gestiona mediante la utilización de un recolector de basura, mecanismo implícito que se encarga de reservar, liberar y compactar espacios de memoria. [21]

Este lenguaje fue desarrollado por la empresa *Sun Microsystems* a principios de la década de los 90. Las aplicaciones Java se encuentran generalmente compiladas en

un *bytecode*, aunque también está permitido la compilación en código máquina nativo. Durante el tiempo de ejecución, el *bytecode* es generalmente interpretado, pero también es posible la ejecución directa por hardware del bytecode mediante un procesador Java. [22]

A continuación podemos observar un gráfico en el que se puede observar la evolución de la tendencia de lenguajes de programación para el desarrollo de software en los últimos diez años, según un estudio realizado por la comunidad TIOBE [23].

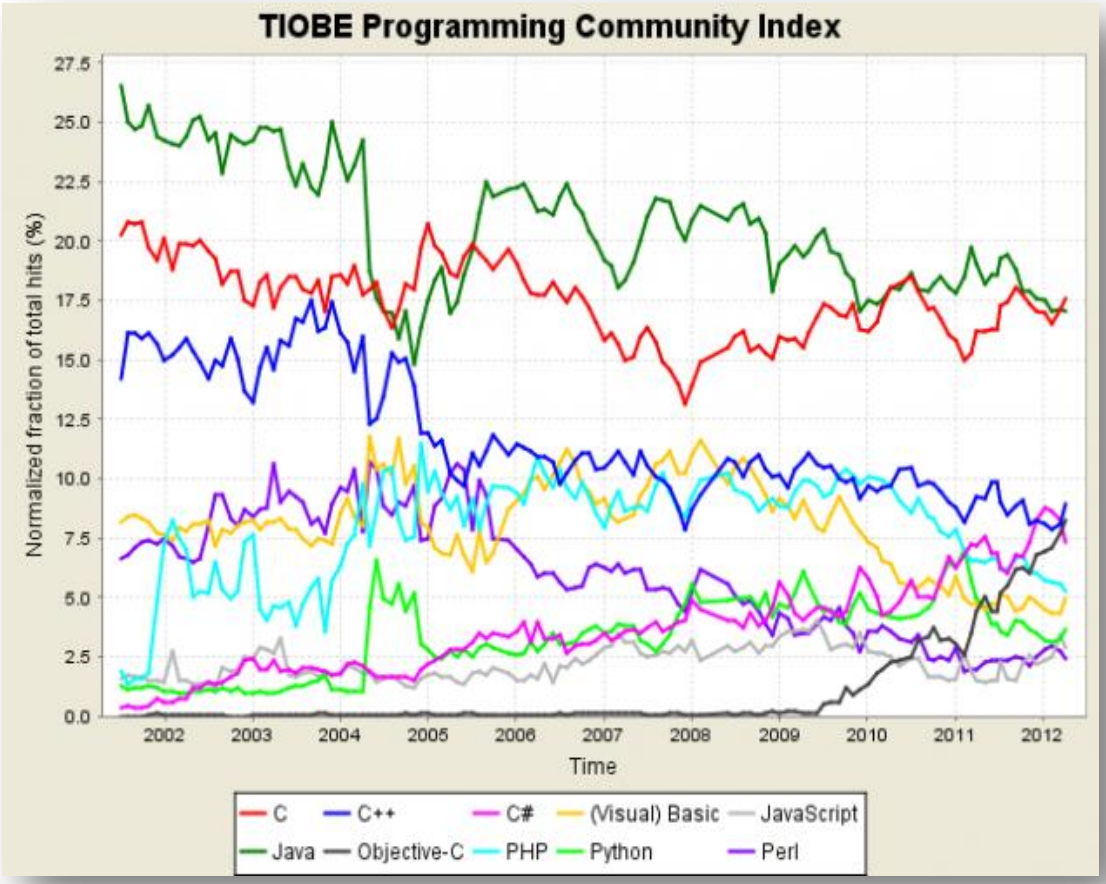


Ilustración 10. Popularidad de lenguajes de programación. (Fuente: TIOBE)

Podemos observar como en el último año el lenguaje C supera en popularidad a Java, aunque es este último el que siempre ha sido más popular a lo largo de los años, excepto en mínimos periodos de tiempo.

Otro de los aspectos más importantes de este lenguaje es la capacidad de portabilidad del software generado, que puede ser ejecutado en diferentes plataformas, gracias a la traducción a código intermedio independiente de la máquina.

Además, *Java* tiene una amplia documentación y diversos entornos de programación. Uno de ellos es el entorno de programación *Eclipse*, cuyas características analizaremos a continuación. [24]

ECLIPSE

Eclipse [25] es uno de los entornos de programación más conocidos para el lenguaje Java. Una de sus principales características es que posee un gran número de *plugins* que convierten a este entorno en uno de los más completos. Su extensa flexibilidad permite adaptarse a múltiples situaciones, pero para ello debe ser convenientemente configurado. En el caso de trabajar con *JSPs*, como se necesitará en este proyecto, hay que aplicar al entorno extensiones y configurarlas para su correcto funcionamiento.

Además, posee un gran soporte de refactorización y ayuda en la codificación, ofreciendo múltiples sugerencias y autocompletando simultáneamente mientras se codifica.

ALTERNATIVAS DE ENTORNOS DE DESARROLLO

Otra alternativa es el entorno de desarrollo libre centrado para Java denominado *NetBeans*, [26] muy similar al entorno descrito anteriormente, este posee sin necesidad de extensiones un diseñador gráfico, pero esta característica no supone mayor relevancia a la hora de decantarse por un *IDE* o por otro, puesto que las posibilidades de la herramienta de diseño son en ocasiones reducidas y únicamente permite visualizar *HTML* básicos.

2.4.2.2 LENGUAJE C# Y ENTORNO DE PROGRAMACIÓN MICROSOFT VISUAL STUDIO

C# es también un lenguaje de programación orientado a objetos desarrollado por Microsoft y estandarizado como parte de su plataforma *.NET*, [27] derivando su sintaxis, al igual que Java, de lenguajes como *C/C++* y utilizando el modelo de objetos similar al de *JAVA*, proveniente de la plataforma *.NET*, aunque incluye mejoras basadas en otros lenguajes.

Es un lenguaje de programación independiente diseñado para generar software únicamente para la anteriormente nombrada plataforma *.NET*. Su gestión de memoria es automática y fácilmente portable gracias a la compilación del software a código intermedio, características similares al anteriormente analizado lenguaje *JAVA*.

El entorno de desarrollo integrado más utilizado que soporta este tipo de lenguaje es *Microsoft Visual Studio*. La limitación más destacable es que se trata de un entorno de programación que únicamente está orientado a sistemas operativos Windows.

2.4.2.3 COMPARACIÓN DEL ESTUDIO Y DECISIÓN DE LA ALTERNATIVA TOMADA

Ambos lenguajes poseen características muy parecidas, la sintaxis de ambos son similares, como muestra se procede a realizar una comparación entre estructuras y sintaxis de ambos lenguajes. [28]

Java — **Program Structure** — **C#**

```
package hello;

public class HelloWorld {
  public static void main(String[] args) {
    String name = "Java";

    // See if an argument was passed from the command line
    if (args.length == 1)
      name = args[0];

    System.out.println("Hello, " + name + "!");
  }
}
```

```
using System;

namespace Hello {
  public class HelloWorld {
    public static void Main(string[] args) {
      string name = "C#";

      // See if an argument was passed from the command line
      if (args.Length == 1)
        name = args[0];

      Console.WriteLine("Hello, " + name + "!");
    }
  }
}
```

Ilustración 11. Diferencia sintáctica entre Java y C# en estructuras de programa

Java — **Functions** — **C#**

```
// Return single value
int Add(int x, int y) {
  return x + y;
}

int sum = Add(2, 3);
```

```
// Return no value
void PrintSum(int x, int y) {
  System.out.println(x + y);
}

PrintSum(2, 3);
```

```
// Return single value
int Add(int x, int y) {
  return x + y;
}

int sum = Add(2, 3);
```

```
// Return no value
void PrintSum(int x, int y) {
  Console.WriteLine(x + y);
}

PrintSum(2, 3);
```

```
// Primitive types and references are always passed by value
void TestFunc(int x, Point p) {
  x++;
  p.x++;      // Modifying property of the object
  p = null;   // Remove local reference to object
}

class Point {
  public int x, y;
}

Point p = new Point();
p.x = 2;
int a = 1;
TestFunc(a, p);
System.out.println(a + " " + p.x + " " + (p == null) );  // 1 3 false
```

```
// Pass by value (default), in/out-reference (ref), and
out-reference (out)
void TestFunc(int x, ref int y, out int z, Point p1, ref Point p2) {
  x++;  y++;  z = 5;
  p1.x++;      // Modifying property of the object
  p1 = null;   // Remove local reference to object
  p2 = null;   // Free the object
}

class Point {
  public int x, y;
}

Point p1 = new Point();
Point p2 = new Point();
p1.x = 2;
int a = 1, b = 1, c;   // Output param doesn't need initializing
TestFunc(a, ref b, out c, p1, ref p2);
Console.WriteLine("{0} {1} {2} {3} {4}",
  a, b, c, p1.x, p2 == null);   // 1 2 5 3 True
```

```
// Accept variable number of arguments
int Sum(int ... nums) {
  int sum = 0;
  for (int i : nums)
    sum += i;
  return sum;
}

int total = Sum(4, 3, 2, 1);   // returns 10
```

```
// Accept variable number of arguments
int Sum(params int[] nums) {
  int sum = 0;
  foreach (int i in nums)
    sum += i;
  return sum;
}

int total = Sum(4, 3, 2, 1);   // returns 10
```

Ilustración 12. Diferencia sintáctica entre Java y C# en funciones

Como se puede observar, la sintaxis es muy similar en todos los casos. Las dos opciones analizadas poseen múltiples similitudes y consiguen aportar prácticamente las mismas funcionalidades, por lo tanto la diferencia en este caso no la marca el lenguaje, sino el entorno de desarrollo utilizado. Ambas alternativas servirían perfectamente para el desarrollo de la aplicación, pero finalmente el entorno escogido será Eclipse junto al lenguaje de programación Java puesto que el equipo de desarrollo posee más experiencia sobre esta opción, a pesar de haber utilizado ambas en anteriores trabajos, además el entorno *Eclipse* no posee la limitación de sistemas operativos que sí presenta la opción de *Microsoft*.

2.4.3 SERVIDORES: TOMCAT Y OTRAS ALTERNATIVAS

En los apartados anteriores hemos realizado el estudio de diversas opciones con el fin de obtener una decisión final para el desarrollo de la herramienta que mostraba el proyecto, finalmente se ha decidido realizar una aplicación de carácter web utilizando las tecnologías ofrecidas por el entorno de desarrollo integrado Eclipse y el lenguaje de programación Java. Es el momento, por lo tanto, de escoger un servidor web con soporte a las tecnologías escogidas previamente.

El servidor web más usado a la hora de trabajar con el lenguaje de programación Java en entornos de desarrollo web es *Tomcat*, [29] haciendo a este servidor una de las implementaciones funcionales de los estándares de *JSP*. Además, gestiona las peticiones de *JSP* recibidas por los servidores más populares como el Apache *HTTP* o el servidor Microsoft Internet Information Server (*IIS*).

Este servidor es desarrollado y mantenido por la fundación de *Apache Software* y un grupo de voluntarios independientes. Es posible acceder libremente a su código fuente y utiliza el contenedor de *servlets* Catalina.

Existe otro popular servidor de aplicaciones de código abierto, en esta ocasión desarrollado por *Sun Microsystems*. Se trata de *GlassFish*, [30] implementando las tecnologías definidas en la plataforma Java EE.

Ambos son bastante similares, las principales diferencias son que *Tomcat* ofrece una tecnología totalmente asentada y, en comparación con otras alternativas (incluido *GlassFish*), es bastante ligero, mientras que GlassFish es relativamente nuevo en el campo de los servidores, a pesar de ser desarrollado por una gran compañía líder encargada del desarrollo del lenguaje Java.

En esta ocasión, todas las aplicaciones que funcionen de forma correcta en uno de los servidores web anteriormente descritos, lo harán sin ningún tipo de problemas o conflictos en el otro. Teniendo en cuenta que la elección anterior fue la de realizar la herramienta mediante el lenguaje Java, esto tampoco supondrá ningún problema en cuanto a compatibilidad se refiere, puesto que *Tomcat* está escrito en Java, lo que lo hace totalmente compatible a la hora de usarlo, y *GlassFish* está desarrollado por *Sun Microsystems*, indicando de esta forma compatibilidad total entre estas tecnologías.

Tomcat no posee contenedor de *EJBs*, pero para el desarrollo de este proyecto no posee relevancia.

En esta ocasión se decide escoger la opción del servidor web *Tomcat*, debido a que su desarrollo posee más años de experiencia, pero se tiene en cuenta la gran opción que también es el servidor de aplicaciones *GlassFish*.

2.4.4 SISTEMA DE GESTIÓN DE LA BASE DE DATOS

El objetivo de un sistema gestor de base de datos es el de manejar un conjunto de datos que serán convertidos en información de carácter relevante, de una forma ordenada, sencilla y clara.

Se trata de una colección de software específico cuya finalidad es realizar el papel de interfaz entre el usuario, las aplicaciones que la utilizan y la base de datos que contiene la información. Siendo esto no visible al usuario final y encargándose en todo momento de la privacidad, la integridad y la seguridad de la información.

Un sistema gestor de Base de Datos permite el almacenamiento, consulta y manipulación de datos que forman parte de una base de datos organizada en uno o varios archivos. De forma general, la base de datos consiste en un conjunto de tablas entre las que quedan establecidas diversas relaciones. Las características fundamentales de un Sistema de Gestión de Base de Datos son las siguientes: [31]

- Mantener de forma independiente el método de almacenamiento y el programa que gestiona los datos (servidor) del programa desde el cual se realizan las consultas (cliente). Así se consigue independencia en la información, de manera que si se debe modificar el esquema, tanto lógico como físico, no es necesario realizar ningún cambio en ninguna de las aplicaciones que estén haciendo uso del sistema gestor de la base de datos.

Ilustración 13. Esquema de un sistema gestor de base de datos

- Permite consultas complejas, cuya resolución se encuentra totalmente optimizada y es expresada con el uso de un lenguaje formal.

- El almacenamiento de la información se realiza de una forma eficiente y oculta, de esta forma el usuario visualiza una estructura bastante diferente a la que presentan los datos en su almacenamiento. De esta forma, ahorran al usuario todo tipo de detalles relacionados con el almacenamiento físico de los datos, de esta forma surgen diferentes niveles de abstracción de la información.

- Garantizan la ausencia de conflictos o problemas de seguridad ante el acceso de varios usuarios autorizados de forma concurrente, solventando problemas de integridad o pérdida de información en accesos simultáneos. Esto es conseguido gracias al sistema de categorización de permisos y división de usuarios en grupos de usuarios.

La utilización de un gestor de base de datos favorece muchos aspectos que serán descritos a continuación: [32]

+ Son capaces de simplificar la programación de equipos de consistencia.

+ Manejan políticas de respaldo de tal forma que son capaces de garantizar que cualquier modificación de la base de datos sucederá siempre de manera consistente.

+ Organizan los datos sin afectar en gran medida al código de las aplicaciones.

+ Disminuyen potencialmente los tiempos de desarrollo.

+ Incrementan la calidad del sistema desarrollado en el caso de que los desarrolladores sean capaces de explotarlos correctamente.

2.4.4.1 MYSQL

Existen numerosos sistemas de gestión de base de datos. Cabe destacar entre los sistemas libres: *PostgreSQL*, *Firebird*, *SQLite*, entre otros; y en cuanto a los sistemas no libres: *Oracle*, *Microsoft SQL Server* y *IBM DB2*. En este apartado se analizará el gestor de base de datos escogido para el desarrollo de este proyecto: *MySQL*, de carácter libre.

MySQL [33] se ofrece como software libre desde enero de 2008 de un esquema de licenciamiento dual. Es uno de los gestores que más atrae a los desarrolladores de aplicaciones web con contenido dinámico gracias a su simplicidad y sus óptimos resultados.

Se trata de uno de los gestores con mejor rendimiento, puesto que posee una gran velocidad a la hora de realizar las operaciones de gestión de la base de datos. Además, teniendo en cuenta sus requerimientos para la elaboración de bases de datos, ofrece un bajo consumo de recursos, pudiéndose ejecutar en equipos no demasiado complejos. Este gestor presenta una fácil instalación y configuración, y soporta gran variedad de sistemas operativos.

Es por estas razones que se ha decidido utilizar dicho sistema de gestión de base de datos para el desarrollo del proyecto. Para ello es necesario configurar el entorno de desarrollo anteriormente escogido: Eclipse, es necesario hacer uso del *driver* java para poder conectar el entorno con la base de datos. El desarrollador de este sistema gestor facilita en esta ocasión el controlador necesario *'mysql-connector-java'* que hace compatible dicho entorno con esta tecnología.

Para la ejecución de operaciones sobre la base de datos desde el lenguaje de programación de Java, además, es necesario hacer uso de la *API Java Database Connectivity*, [34] conocida por sus siglas *JDBC*, que ofrece el paquete *java.sql*.

3. ANÁLISIS DEL SISTEMA

En esta fase, el objetivo prioritario es identificar cada una de las necesidades que deben ser satisfechas al desarrollar el sistema final. Esto se hará efectivo mediante la obtención de requisitos que deben estar presentes en el sistema de información final, con la finalidad de dar solución a las necesidades identificadas con anterioridad.

Este proceso de extracción de la información debe ser realizado entre los ingenieros del sistema y los usuarios del mismo, y también dará lugar a un estudio detallado de las operaciones que deben ser implementadas en el sistema de información, dando lugar de esta forma al análisis de casos de uso.

Al finalizar esta primera etapa en el desarrollo del producto se consiguen los requisitos, tanto funcionales como no funcionales, y su modelado mediante los casos de uso, desarrollados estos últimos en diagramas y también descritos textualmente. Todo esto facilita la comprensión del sistema haciendo que pueda ser diseñado cumpliendo con todos los objetivos y cubriendo las necesidades de forma satisfactoria.

DEFINICIÓN DEL SISTEMA

La idea principal está basada en la creación de una herramienta mediante la cual sea posible crear un escenario de juegos que después será visualizado mediante la aplicación Unity 3D.

Ilustración 14. Escenario 3D generado por el motor Unity 3D

Los escenarios que deben ser generados están compuestos por diversas escenas, cada una de diferente tamaño, que deben ser recorridas por un personaje en las que se encontrará con diversas entidades (objetos).

El nivel de edición de dichas escenas por parte de la aplicación debe tener en cuenta tanto el tamaño de dichas escenas como todo lo que las compone. Un ejemplo son las texturas de las paredes y del suelo, así como la posición de todo ello en el escenario. Al igual sucede con las entidades, deben ser posicionadas sobre distintas partes del escenario y establecer de qué tipo se trata.

Toda esta información debe almacenarse en un documento XML, que posteriormente será utilizado para la generación del escenario.

A continuación se profundiza más detalladamente en todas y cada una de las funcionalidades de esta herramienta, analizando el sistema mediante la identificación de requisitos y la especificación de los casos de uso.

3.1 IDENTIFICACIÓN DE REQUISITOS

En esta sección la finalidad es detallar de una manera clara y concisa cada uno de los objetivos que presenta el proyecto, así como todas sus características. Dicha información ha sido concretada en las diferentes reuniones establecidas con el tutor del proyecto, que en este caso realiza el rol de cliente del producto. Como veremos a continuación, algunos requisitos iniciales han sido modificados o directamente eliminados, puesto que a lo largo de todas las reuniones con el cliente se determinó que alguno de las funcionalidades no eran posibles llevarlas a cabo, o debía ser realizado de otra forma, en su mayoría con el fin de mejorar y optimizar la aplicación.

3.1.1 TABLA DE REQUISITOS Y NOMENCLATURA

A continuación se muestra la plantilla utilizada para la identificación de los requisitos, así como la nomenclatura utilizada para todos ellos.

IDENTIFICADOR:
NOMBRE:
NECESIDAD:
PRIORIDAD:
VERIFICABILIDAD:
ESTABILIDAD:
DESCRIPCIÓN:
MODIFICACIONES:

Tabla 1. Plantilla de requisitos

- Identificador: Único para cada requisito. Consta de los siguientes atributos:
  - **RQ-F ID** ó **RQ-NF ID** siendo
    - RQ-F: Requisito Funcional.
    - RQ-FN: Requisito No Funcional.
    - ID: Número de identificación del requisito.
- Necesidad: Indica el nivel de prioridad de inclusión del requisito en el sistema.
- Prioridad: Indica el grado de importancia del orden de implementación. Pudiendo determinar entre los siguientes niveles:
  - Alto
  - Medio
  - Bajo
- Verificabilidad: Indica el grado de comprobación de su implementación de forma correcta, determinada entre los siguientes niveles:
  - Alto
  - Medio
  - Bajo
- Estabilidad: Determina si el requisito está sujeto o no a modificaciones.
- Descripción: Breve explicación del requisito.
- Modificaciones: Establece los cambios que ha sufrido el requisito a lo largo del desarrollo, si es que han ocurrido modificaciones.

3.1.2 REQUISITOS FUNCIONALES

Requisitos encargados de especificar todas las funcionalidades que el sistema debe satisfacer.

IDENTIFICADOR: RQ-F 01	
NOMBRE: Determinación del tamaño del escenario	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alta	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación permite al usuario determinar tamaño y forma del escenario mediante la colocación de las escenas creadas por el mismo.
MODIFICACIONES:	En primera instancia se decidió asignar el tamaño del escenario desde un primer momento, antes de la creación de escenas, basándose en una cuadrícula de tamaño máximo 20x20.

	En reuniones posteriores se decide no restringir el tamaño máximo, además la asignación del tamaño del escenario no se realiza directamente, sino que queda determinada por las escenas creadas del mismo.

Tabla 2. Requisito de software funcional 1

IDENTIFICADOR: RQ-F 02	
NOMBRE: Selección escena en escenario	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Media	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación mostrará el escenario con el tamaño final escogido permitiendo seleccionar las casillas para editar cada escena.
MODIFICACIONES:	La modificación del RQ-F 01 altera a su vez este requisito, el tamaño de la escena no se selecciona sobre el escenario, dando total libertad al usuario en dicha selección de tamaño. El tamaño y la forma del escenario quedan definidos con la colocación de las escenas.

Tabla 3. Requisito de software funcional 2

IDENTIFICADOR: RQ-F 03	
NOMBRE: Introducción nombre de escena	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alta	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación solicitará al usuario la introducción en un campo de entrada del nombre que identifique la escena.

Tabla 4.Requisito de software funcional 3

IDENTIFICADOR: RQ-F 04
NOMBRE: Selección del tamaño de escena
NECESIDAD: Esencial
PRIORIDAD: Alta
VERIFICABILIDAD: Alta

ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación solicitará al usuario la introducción del alto y ancho de la escena, representando dichos valores el número de *tails* de ambos atributos.

Tabla 5.Requisito de software funcional 4

IDENTIFICADOR: RQ-F 05	
NOMBRE: Creación de nuevo escenario	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alta	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación permitirá al usuario crear un nuevo escenario al inicio de creación de una nueva escena. Esta será la primera escena de dicho escenario. La aplicación solicita como identificación del mismo, al usuario, un nombre mediante un campo de entrada.

Tabla 6.Requisito de software funcional 5

IDENTIFICADOR: RQ-F 06	
NOMBRE: Selección de un escenario ya existente	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alta	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación permitirá al usuario seleccionar un escenario previamente creado, al inicio de creación de una nueva escena. Esta escena pasará a formar parte de la composición del escenario junto con las ya existentes. La aplicación permite seleccionar la identificación del escenario al usuario mediante un campo select.

Tabla 7.Requisito de software funcional 6

IDENTIFICADOR: RQ-F 07
NOMBRE: Selección de paredes en escena
NECESIDAD: Esencial
PRIORIDAD: Alta
VERIFICABILIDAD: Alta

ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación permitirá seleccionar las paredes que componen la escena seleccionada, siendo las posibilidades por cada *tail*: Pared norte, pared sur, pared este, pared oeste. El máximo está establecido en las 4 paredes, el mínimo ninguna. Se realizará mediante la visualización de la escena en todas las *tails* que la componen.

Tabla 8.Requisito de software funcional 7

IDENTIFICADOR: RQ-F 08	
NOMBRE: Selección del tipo de escena	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Media	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación solicitará al usuario la selección del tipo de escena a través de un campo *select*. Las opciones a escoger por el usuario son: Jardín, Cancha de Básquet, Campo de futbol, Jardín, Clase y Pasillo.
MODIFICACIONES:	Esta categorización de escenas finalmente es eliminada, de manera que la aplicación no limita el tipo de escena que se está creando en dicho momento.

Tabla 9.Requisito de software funcional 8

IDENTIFICADOR: RQ-F 09	
NOMBRE: Selección de enlaces en escena	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Media	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación permitirá seleccionar los enlaces que compondrán una escena, basándose en las paredes introducidas en la misma. Las opciones a escoger de los enlaces en las paredes son: abierto y cerrado. Se realizará mediante la visualización de la escena en todas las casillas que la componen.
MODIFICACIONES:	Finalmente se decide tratar dichos enlaces como entidades, tal como se explica en el RQ-F 07.

Tabla 10.Requisito de software funcional 9

IDENTIFICADOR: RQ-F 10	
NOMBRE: Definición de conjuntos de tipos de entidades	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Media	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación presentará un conjunto de entidades para asociar a las escenas del mapa. La definición de los mismos estará determinada por los siguientes términos: - Nombre, que se escogerá a través de un campo select - Posición Este requisito explota en los RQ-F 11 y 12

Tabla 11.Requisito de software funcional 10

IDENTIFICADOR: RQ-F 11	
NOMBRE: Selección de entidades en escena (Nombre)	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alta	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación permitirá seleccionar las entidades que compondrán una escena. Se realizará mediante la visualización de la escena en todas las casillas que la componen a través de un campo select.
MODIFICACIONES:	Son añadidas como entidades, los enlaces de las paredes de las escenas. Incorporando de esta forma el RQ-F 0X

Tabla 12.Requisito de software funcional 11

IDENTIFICADOR: RQ-F 12
NOMBRE: Selección de entidades en escena (Posición)
NECESIDAD: Esencial
PRIORIDAD: Alta
VERIFICABILIDAD: Alta
ESTABILIDAD: Durante toda la vida del software

DESCRIPCIÓN:	La aplicación permitirá seleccionar la posición de las entidades que compondrán una escena. Se realizará mediante tres campos de entrada que permitirán al usuario el establecimiento de las coordenadas X,Y,Z de cada una.

Tabla 13.Requisito de software funcional 12

IDENTIFICADOR: RQ-F 13	
NOMBRE: Visualización de la escena (Vista cenital)	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alta	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	- La aplicación permitirá la visualización de la escena durante su edición mediante una vista cenital, representando casillas, paredes y entidades de la misma.

Tabla 14.Requisito de software funcional 13

IDENTIFICADOR: RQ-F 14	
NOMBRE: Selección de textura de la pared	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alta	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación permitirá seleccionar la textura de la pared en la edición de la escena, creando las paredes de cada *tail* según la textura seleccionada en ese instante.

Tabla 15.Requisito de software funcional 14

IDENTIFICADOR: RQ-F 15	
NOMBRE: Selección de textura del suelo	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alta	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación permitirá seleccionar la textura del suelo en la edición de la escena, creando el suelo de las *tails* según la textura seleccionada en ese instante.

Tabla 16.Requisito de software funcional 15

IDENTIFICADOR: RQ-F 16	
NOMBRE: Selección de altura del suelo	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alta	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación permitirá seleccionar la altura del suelo en la edición de la escena, seleccionando el suelo de la casilla y escogiendo mediante un campo *select* el nivel de altura de la misma.

Tabla 17.Requisito de software funcional 16

IDENTIFICADOR: RQ-F 17	
NOMBRE: Posición de la escena en el escenario	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Media	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación permitirá al usuario posicionar la escena creada en el escenario seleccionado al principio del proceso, visualizando todas las demás escenas del mismo en el caso de que se trate de un escenario ya existente.
MODIFICACIONES:	En un principio se decidió que la aplicación mostraría el escenario con el tamaño final escogido permitiendo seleccionar las casillas para editar cada escena. En posteriores reuniones se determinó que el tamaño de la escena no se selecciona sobre el escenario, dando total libertad al usuario en dicha selección de tamaño. El tamaño y la forma del escenario quedan definidos con la colocación de las escenas, tal como queda determinado en el **RQ-F 01**

Tabla 18.Requisito de software funcional 17

IDENTIFICADOR: RQ-F 18	
NOMBRE: Fases de edición del escenario	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alta	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación mostrará durante todo el proceso de edición de un escenario el paso en el que el usuario se encuentra, posibilitando la vuelta atrás a una fase ya realizada.

Tabla 19.Requisito de software funcional 18

IDENTIFICADOR: RQ-F 19	
NOMBRE: Visualización del escenario final	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alta	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación permitirá visualizar el escenario final una vez añadidas las escenas por parte del usuario. Se tratará de una vista cenital con todas las escenas del escenario en cuestión.

Tabla 20.Requisito de software funcional 19

IDENTIFICADOR: RQ-F 20	
NOMBRE: Guardar escenario	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Media	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación permitirá guardar el escenario realizado a partir de todas las escenas creadas.

Tabla 21.Requisito de software funcional 20

IDENTIFICADOR: RQ-F 21	
NOMBRE: Generación de archivos	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Media	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación generará un fichero XML con la información del escenario, que podrá ser utilizado en la herramienta Unity 3D para la generación de dicho escenario. Las restricciones y formato del documento aparecen descritas en el **RQ-NF 07** y en el anexo II del documento.

Tabla 22.Requisito de software funcional 21

3.1.2 REQUISITOS NO FUNCIONALES

Estos requisitos representan restricciones, necesidades o condiciones de las funcionalidades del sistema, o algunos aspectos de la seguridad del mismo.

IDENTIFICADOR: RQ-NF 01	
NOMBRE: Aplicación Web	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alto	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La herramienta debe ser una aplicación web, a la que el usuario pueda acceder a través del acceso a un servidor web mediante la utilización de un navegador.

Tabla 23.Requisito de software no funcional 1

IDENTIFICADOR: RQ-NF 02	
NOMBRE: Interfaz simple e intuitiva	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Media	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación deberá presentar una interfaz intuitiva y sencilla en todas las fases del proceso de creación del escenario, con el fin de que la interacción para el usuario sea la más cómoda posible.

Tabla 24.Requisito de software no funcional 2

IDENTIFICADOR: RQ-NF 03	
NOMBRE: Control de errores en la escena	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alto	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación controlará que no se establezca un tamaño incorrecto en la escena para poder generar su visualización en la siguiente fase del proceso.

Tabla 25.Requisito de software no funcional 3

IDENTIFICADOR: RQ-NF 04	
NOMBRE: Datos previamente cargados en selección de escenarios	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alto	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación presentará en el campo de escenarios ya existentes un desplegable con todos los escenarios creados hasta el momento.

Tabla 26.Requisito de software no funcional 4

IDENTIFICADOR: RQ-NF 05	
NOMBRE: Datos necesarios para creación de nueva escena	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alto	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	El usuario debe completar los siguientes datos con el fin de crear una nueva escena: - Nombre de la escena - Alto de la escena - Ancho de la escena - Desplegable de selección de escenario ó Nombre de nuevo escenario.

Tabla 27.Requisito de software no funcional 5

IDENTIFICADOR: RQ-NF 06	
NOMBRE: Manual de usuario	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alto	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	Deberá proporcionarse junto a la aplicación un manual de usuario.

Tabla 28.Requisito de software no funcional 6

IDENTIFICADOR: RQ-NF 07	
NOMBRE: Formato de documento XML generado	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alto	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	El documento XML generado por la aplicación seguirá un formato específico facilitado por parte del cliente. En el mismo queda definida toda la información relacionada con el escenario, los datos que aparecen en dicho fichero son los siguientes:

	- Información de la escena que compone en escenario Suelos de la escena: Posición, tamaño y textura. Paredes de la escena: Posición, tamaño y textura. Definición de entidades: Tipo de entidad y posición. Punto de aparición del personaje. En el anexo II de este documento se adjunta un ejemplo de fichero XML generado con la aplicación.

Tabla 29.Requisito de software no funcional 7

IDENTIFICADOR: RQ-NF 08	
NOMBRE: Compatibilidad con los principales navegadores	
NECESIDAD: Esencial	
PRIORIDAD: Alta	
VERIFICABILIDAD: Alto	
ESTABILIDAD: Durante toda la vida del software	
DESCRIPCIÓN:	La aplicación debe ser compatible y totalmente funcional con los navegadores: Firefox y Chrome.

Tabla 30.Requisito de software no funcional 8

3.3 CASOS DE USO

A continuación se detallarán los casos de uso, describiendo todas las posibles interacciones entre el usuario del sistema y la aplicación con el fin de alcanzar un objetivo. De esta manera se proporcionan diversos escenarios que indican la sucesión de respuestas surgidas ante las acciones iniciadas por los usuarios sobre el sistema.

Esta especificación de casos de uso facilita la comprensión del sistema y expresa la intención con la que el usuario interactuará con la aplicación, profundizando así en dichas necesidades, y que complementándose con los requisitos anteriormente descritos, establece una firme base a la hora de comenzar con la fase de diseño.

En el siguiente apartado queda determinada la notación utilizada, basada en el Lenguaje Unificado de Modelado (UML) y los diferentes elementos que componen cada diagrama de casos de uso.

3.3.1 DIAGRAMAS DE CASOS DE USO

Los elementos de los que consta cada diagrama de casos de uso son los siguientes:

- Actor: se trata del rol tomado por la entidad que actúa de forma externa al sistema, interactuando con el mismo con el objetivo de satisfacer una necesidad mediante las funcionalidades que el sistema presenta.

  Queda definido como 'rol', puesto que no necesariamente debe tratarse de una entidad humana, puesto que realmente lo que define son las acciones que se realizan sobre el sistema, pudiéndose tratar de esta forma tanto de una persona como de otra entidad externa o abstracta.

  En el diagrama de casos de uso quedará representado de la siguiente forma:

Ilustración 15. Representación de actor en diagrama de casos de uso

- Caso de uso: es la operación realizada tras un evento, siendo este realizado por petición del actor u otra entidad externa, o por acción de otro caso de uso.

  La representación de este elemento en los diagramas es la siguiente:

Ilustración 16. Representación de Caso de uso en diagramas de casos de uso

- Tipos de relaciones: el vínculo entre actor y caso de uso puede ser de diversos tipos.

  - *Relación de asociación*: relación básica entre casos de uso, o un actor y un caso de uso que representa la invocación de una operación. En el diagrama queda representado por una línea de unión simple:

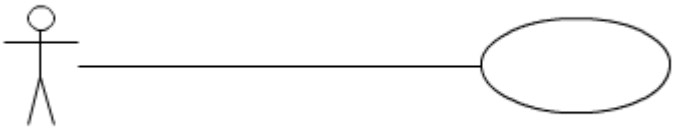


Ilustración 17. Representación de relación de asociación en diagramas de casos de uso

- *Relación de inclusión*: interacción entre casos de uso, donde la invocación de una operación de uno depende del resultado del caso de uso incluido. La representación en este caso queda determinada mediante una flecha con línea discontinua y una etiqueta *'include'*:

Ilustración 18. Relación de relación de inclusión en diagrama de casos de uso

- Escenario: establece los límites del sistema.

  En el diagrama de casos de uso, el escenario queda representado de la siguiente forma:

Ilustración 19. Representación de escenario en diagramas de casos de uso

Para la especificación de los diagramas de casos de uso del sistema de generación de escenarios observamos que el elemento del actor queda definido por el rol que toma el usuario de la aplicación.

- **Usuario de la aplicación de generación de escenarios:** representa al usuario que interactúe con la aplicación de generación de escenarios, pudiendo acceder a todas las funcionalidades del sistema de generación de escenarios.

Además se va a complementar cada diagrama de caso de uso mostrado con una descripción textual del mismo, consiguiendo así una explicación más detallada y formal de cada uno. También, aparecerán las condiciones necesarias para su correcta realización, los roles que son capaces de desarrollarlos y el escenario en el cual cada caso de uso debe operar, estableciendo conjuntamente el estado del sistema en cada ocasión.

En primer lugar, se presenta la plantilla utilizada para la descripción textual de los casos de uso y la nomenclatura utilizada:

IDENTIFICADOR		NOMBRE	
Objetivo			
Actor			
Precondiciones			
Postcondiciones			
Escenario básico			

Tabla 31. Plantilla de descripción textual de casos de uso

- Identificador: CU E0 – 00
  - Siendo CU (Caso de Uso), E0 (Escenario 0) y 00 el número de identificación del caso de uso.
- Objetivo: Breve descripción del objetivo.
- Actor: Actor que interacciona con el caso de uso en cuestión.
- Precondiciones: Condiciones que deben cumplirse de forma previa para realizar la operación.
- Postcondiciones: Estado del sistema después de la realización de la operación.
- Escenario básico: Limites del sistema, descripción por fases para el cumplimiento del caso de uso.

ESPECIFICACIÓN Y DESCRIPCIÓN DE LOS CASOS DE USO

DIAGRAMA DE CASO DE USO GENERAL:

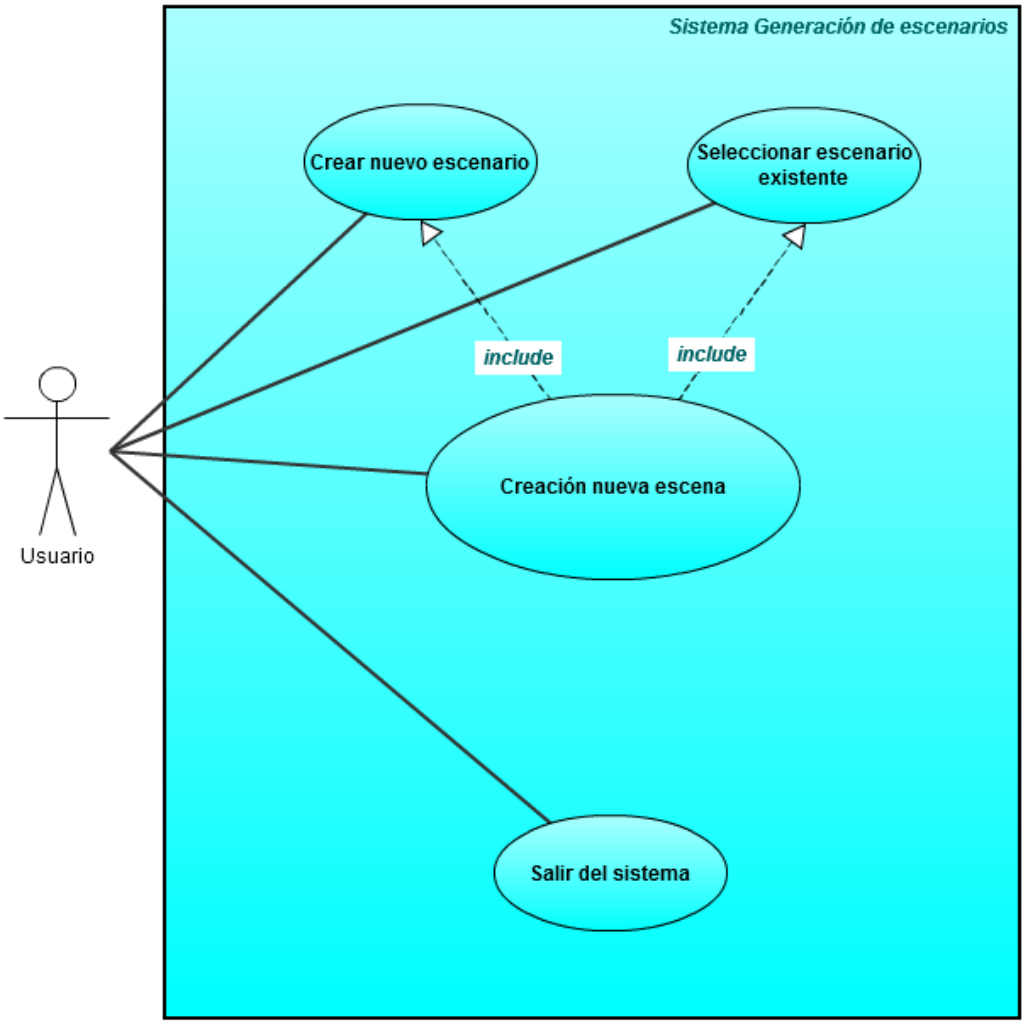


Ilustración 20. Diagrama de caso de uso general

En este diagrama general del sistema se observa el rol que desempeña el usuario de la aplicación encargado de generar escenarios, como único actor en el escenario. Dicho escenario representa el sistema en su totalidad y abarca todos los casos de uso generales como son, la creación de un escenario nuevo o selección de uno ya existente, la creación de la nueva escena (definición y composición, tanto de ella misma como en el escenario) y la opción de abandonar el sistema.

DESCRIPCIÓN TEXTUAL DEL CASO DE USO GENERAL:

IDENTIFICADOR	CU E1 - 01	NOMBRE	CREAR NUEVO ESCENARIO
Objetivo	Creación de un nuevo escenario vacío, para insertar en el mismo nuevas escenas.		
Actor	Usuario de la aplicación de generación de escenarios.		
Precondiciones	---		
Postcondiciones	---		
Escenario básico	1. Se accede a la aplicación 2. Se accede a la pantalla de creación de una nueva escena 3. En la misma seleccionamos la opción de utilizar un nuevo escenario, añadiendo un nombre al mismo.		

Tabla 32. Descripción textual de Caso de uso E1 - 01

IDENTIFICADOR	CU E1 - 02	NOMBRE	SELECCIONAR ESCENARIO EXISTENTE
Objetivo	Selección de un escenario previamente creado, con el fin de añadir una escena nueva al mismo.		
Actor	Usuario de la aplicación de generación de escenarios.		
Precondiciones	Debe haberse creado el escenario anteriormente.		
Postcondiciones	---		
Escenario básico	1. Se accede a la aplicación 2. Se accede a la pantalla de creación de una nueva escena 3. En la misma seleccionamos la opción de utilizar un escenario ya existente mediante el campo select, donde se desplegarán todos los escenarios creados hasta el momento por la aplicación.		

Tabla 33. Descripción textual de Caso de uso E1 - 02

IDENTIFICADOR	CU E1 - 03	NOMBRE	SALIR DE LA APLICACIÓN
Objetivo	Abandonar la aplicación y las operaciones realizadas en la misma.		
Actor	Usuario de la aplicación de generación de escenarios.		
Precondiciones	---		
Postcondiciones	---		
Escenario básico	En cualquier fase del proceso de creación de escenarios el usuario puede abandonar la aplicación de forma inmediata.		

Tabla 34. Descripción textual de Caso de uso E1 - 03

El cuarto caso de uso 'Creación de escena' de este escenario es explotado en varios casos de uso, para facilitar la comprensión del mismo será desglosada también su descripción textual en el escenario siguiente.

DIAGRAMA DE CASO DE USO DE CREACIÓN DE LA NUEVA ESCENA

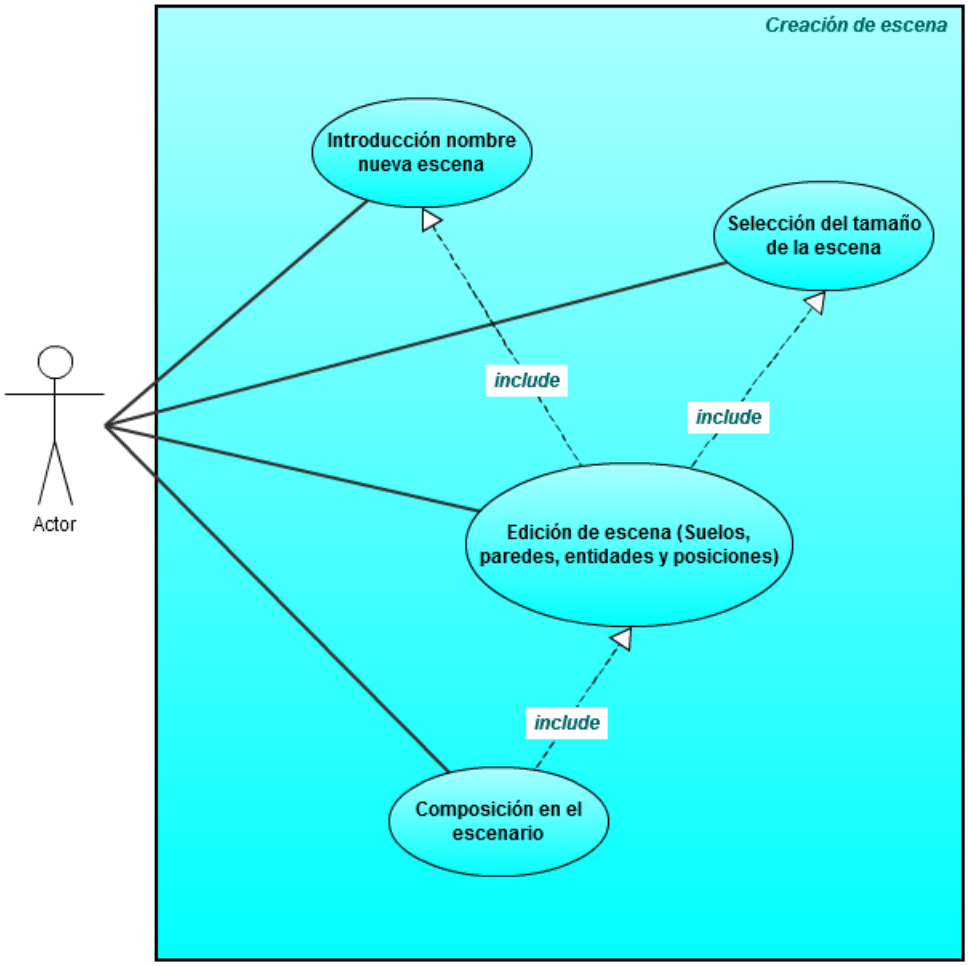


Ilustración 21. Diagrama de caso de uso de creación de nueva escena

En este diagrama, explotamos el caso de uso de creación de la nueva escena limitando el sistema como queda representado mediante este escenario, obteniendo los demás casos de uso presentes en el diagrama: Introducción del nombre, selección del tamaño, edición de la escena (que abarca todas las operaciones relacionadas con el diseño de la misma: paredes, suelos y entidades, junto con las posiciones, tamaños y texturas de las mismas) y por último, la composición del escenario (nuevo o ya existente) con la nueva escena recién creada.

DESCRIPCIÓN TEXTUAL DEL CASO DE USO DE CREACIÓN DE LA NUEVA ESCENA

IDENTIFICADOR	CU E2 - 01	NOMBRE	INTRODUCCIÓN NOMBRE ESCENA
Objetivo	Introducción de un nombre para la nueva escena a crear.		
Actor	Usuario de la aplicación de generación de escenarios.		
Precondiciones	Seleccionar un escenario o crear uno nuevo.		
Postcondiciones	---		
Escenario básico	1. Se accede a la aplicación 2. Se accede a la pantalla de creación de una nueva escena 3. En la misma escogemos entre la creación de un nuevo escenario o la elección de uno previamente creado. 4. Se introduce en un campo de entrada de texto el nombre para la nueva escena.		

Tabla 35. Descripción textual de Caso de uso E2 - 01

IDENTIFICADOR	CU E2 - 02	NOMBRE	SELECCIÓN TAMAÑO DE ESCENA
Objetivo	Selección de un ancho y alto de casillas para la nueva escena.		
Actor	Usuario de la aplicación de generación de escenarios.		
Precondiciones	Seleccionar un escenario o crear uno nuevo. Haber nombrado la nueva escena		
Postcondiciones	---		
Escenario básico	1. Se accede a la aplicación 2. Se accede a la pantalla de creación de una nueva escena 3. En la misma escogemos entre la creación de un nuevo escenario o la elección de uno previamente creado. 4. Se introduce en un campo de entrada de texto el nombre para la nueva escena. 5. Se introduce el alto y el ancho para la escena. Estos valores equivalen al número de casillas de la misma.		

Tabla 36. Descripción textual de Caso de uso E2 - 02

IDENTIFICADOR	CU E2 - 03	NOMBRE	EDICIÓN DE ESCENA
Objetivo	Edición de la composición de la escena nueva.		
Actor	Usuario de la aplicación de generación de escenarios.		
Precondiciones	Seleccionar un escenario o crear uno nuevo. Haber nombrado la nueva escena con un tamaño determinado.		
Postcondiciones	---		
Escenario básico	1. Se accede a la aplicación 2. Se accede a la pantalla de creación de una nueva escena 3. En la misma escogemos entre la creación de un nuevo escenario o la elección de uno previamente creado. 4. Se introduce en un campo de entrada de texto el nombre para la nueva escena. 5. Se introduce el alto y el ancho para la escena. Estos valores equivales a *tails* de la misma. 6. En esta fase del proceso se escogen las diferentes texturas, alturas, posiciones de entidades y lo demás relacionado con la composición de la escena.		

Tabla 37. Descripción textual de Caso de uso E2 - 03

IDENTIFICADOR	CU E2 - 04	NOMBRE	COMPOSICIÓN EN ESCENARIO
Objetivo	Edición de la composición de la escena nueva.		
Actor	Usuario de la aplicación de generación de escenarios.		
Precondiciones	Seleccionar un escenario o crear uno nuevo. Haber nombrado la nueva escena con un tamaño determinado.		
Postcondiciones	---		
Escenario básico	1. Se accede a la aplicación 2. Se accede a la pantalla de creación de una nueva escena 3. En la misma escogemos entre la creación de un nuevo escenario o la elección de uno previamente creado. 4. Se introduce en un campo de entrada de texto el nombre para la nueva escena. 5. Se introduce el alto y el ancho para la escena. Estos valores equivales a tails de la misma. 6. En esta fase del proceso se escogen las diferentes texturas, alturas, posiciones de entidades y lo demás relacionado con la composición de la escena. 7. Se escoge la posición de la escena recién creada en el escenario.		

Tabla 38. Descripción textual de Caso de uso E2 - 04

Con el objetivo de facilitar la comprensión del escenario básico de estos dos últimos casos de uso, se desglosará el paso nº 6 del mismo a continuación.

Fase 1	**Definición de suelos y paredes - Textura**
Descripción	La escena presenta el número de casillas escogido por el usuario en el paso anterior. Puede ser definida la textura de cada una de estas casillas, o tails, por parte del usuario. La forma de realizar esto es seleccionando la textura del desplegable y clicando sobre la casilla del suelo o pared deseada. Se observa que la textura de la misma ha sido modificada.

Tabla 39. Desglose Casos de uso E2-03 y 04

Fase 2	**Definición de suelos - Altura**
Descripción	La escena presenta el número de casillas escogido por el usuario en el paso anterior. Puede ser definida la altura de cada una de estas casillas, o tails, por parte del usuario. La forma de realizar esto es seleccionando el nivel de altura del desplegable y clicando sobre la casilla del suelo deseada. Se observa que aparece el nivel de altura encima de la casilla en la que será aplicada en la generación del escenario.

Tabla 40. Desglose Casos de uso E2-03 y 04 II

Fase 3	Definición de entidades
Descripción	Se selecciona la casilla donde se desea establecer la entidad, escogiendo el tipo de la misma mediante un desplegable, y se determina la posición a través de 3 campos de entrada, que serán las coordenadas X, Y y Z. Para que todo esto suceda se debe activar un campo *check* en el apartado de las entidades.

Tabla 41. Desglose Casos de uso E2-03 y 04 III

DIAGRAMA DE CASO DE USO DEL PROPIO SISTEMA:

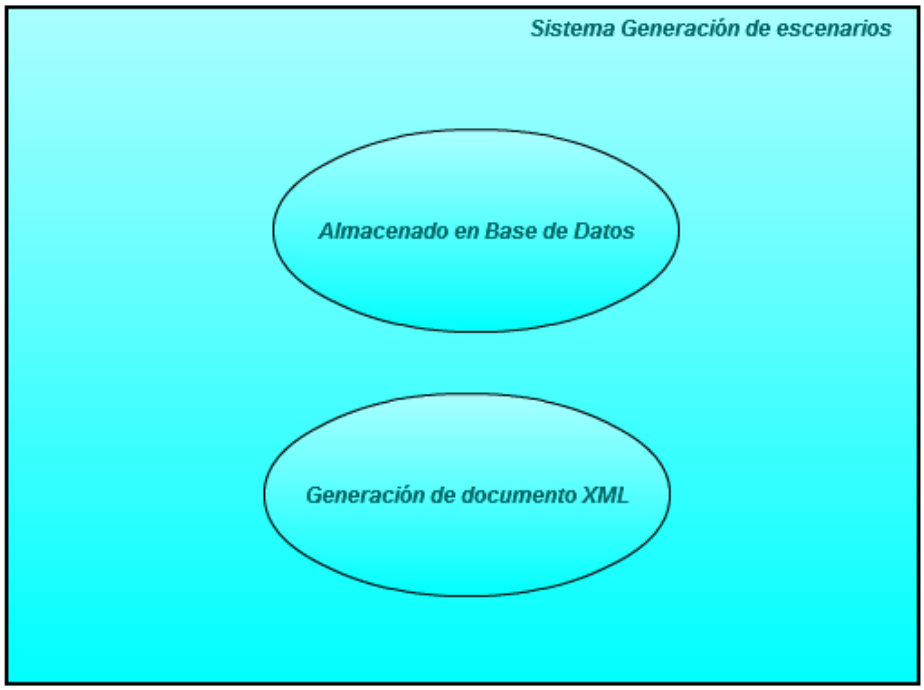


Ilustración 22. Diagrama de caso de uso del propio sistema

Por último vemos este escenario de la totalidad del sistema, pero en esta ocasión sin contar con ningún rol que actúe sobre el mismo. Podría entenderse como el resultado de las operaciones ocurridas con éxito en el diagrama general presentado anteriormente (Diagrama de caso de uso general). De esta forma la aplicación almacena la información en una base de datos y genera un documento XML que será utilizado en la herramienta Unity 3D para la generación del escenario creado.

DESCRIPCIÓN TEXTUAL DEL CASO DE USO DEL PROPIO SISTEMA:

IDENTIFICADOR	CU E3 - 01	NOMBRE	ALMACENADO EN BBDD
Objetivo	Almacenado de información de escenarios en la base de datos.		
Actor	---		
Precondiciones	---		
Postcondiciones	---		
Escenario básico	En diversas situaciones en el proceso de creación de escenarios la aplicación almacena la información en una base de datos.		

Tabla 42. Descripción textual de Caso de uso E3 - 01

IDENTIFICADOR	CU E3 - 02	NOMBRE	GENERACIÓN DE XML
Objetivo	Generación de documento XML con la información necesaria para que la herramienta Unity 3D pueda generar el escenario.		
Actor	---		
Precondiciones	Proceso de creación de escena finalizado correctamente.		
Postcondiciones	---		
Escenario básico	Al finalizar el proceso de creación de un escenario la aplicación genera un documento XML con información del mismo.		

Tabla 43. Descripción textual de Caso de uso E3 - 02

4. DISEÑO DEL SISTEMA

El objetivo de la etapa de diseño, es la definición de la arquitectura del sistema, tanto hardware como software, y la especificación detallada de sus distintos componentes, así como del entorno tecnológico necesario para dar soporte s du funcionamiento.

En esta fase del proceso de desarrollo nos basaremos en toda la información obtenida en la etapa del análisis del sistema, con el fin de dar solución a todas las necesidades que el usuario ha detallado. El objetivo es utilizar dichas bases formadas anteriormente para generar una solución que pueda ser implementada de la forma más satisfactoria, cumpliendo los requisitos analizados.

4.1 VISIÓN GENERAL DE LA SOLUCIÓN

En este apartado se muestra una visión general del producto que se desea implementar, y el escenario donde se va a producir su ejecución. Con ello comprenderemos el funcionamiento del sistema, para después explicar la arquitectura en la que se basa la aplicación.

Existe un rol que será tomado por el usuario de la aplicación de generación de escenarios, será el encargado de interactuar con el sistema en su totalidad. El usuario de la aplicación será un educador que generará un escenario para un determinado juego educativo.

Esta aplicación genera la información necesaria para interactuar con otra herramienta encargada de transferir estos datos a la aplicación Unity 3D, con el objetivo de generar el escenario en 3 dimensiones. Esta información es almacenada en un documento XML y, gran parte de ella, simultáneamente guardada en una base de datos.

Como podemos observar, sólo es necesario definir un único rol para lograr la finalidad completa deseada con el desarrollo de la aplicación. En la siguiente ilustración se muestra dicha interacción entre el usuario y las demás aplicaciones de una manera simplificada.

Ilustración 23. Visión general del proceso de interacción del usuario

4.2 ARQUITECTURA DEL SISTEMA

La elaboración del diseño de la aplicación web de generación de escenarios ha sido basada en el modelo de arquitectura Cliente/Servidor, una de las arquitecturas de aplicación más utilizadas en estos casos.

En primer lugar se detallarán ciertos aspectos sobre esta arquitectura, para después contar ciertas ventajas de su utilización teniendo en cuenta el análisis de este

sistema y por qué ha resultado ser la arquitectura escogida para el desarrollo del mismo.

4.2.1 ARQUITECTURA CLIENTE/SERVIDOR

Esta tecnología Cliente/Servidor basa su modelo en la repartición de tareas, el procesamiento cooperativo de los datos a través de un conjunto de procesadores, donde demandantes de recursos, denominados clientes, solicitan a los servidores estos requerimientos.

Es la arquitectura más utilizada en la realización de Sistemas Distribuidos, permitiendo a los usuarios obtener acceso inmediato a los datos de una forma limpia, y puede ser ejecutada en sistemas multiusuario a través de redes, o también en un único ordenador.

La característica más fundamental y básica en este tipo de arquitecturas es la definición de Servicio, puesto que se trata de la unidad básica de diseño, donde el cliente es el que los demanda al proveedor, que es el servidor. Esto inicia un proceso de recursos compartidos, puesto que gran parte de los demandantes de estos servicios, comparten los mismos servidores, utilizando de esta forma los mismos recursos físicos y lógicos unos con otros. [35]

Existe una fuerte transparencia en cuanto a la localización de los servidores y también de los clientes, la situación de los recursos que utiliza cada cliente no es desvelada.

Un sistema que emplea esta arquitectura presenta el siguiente esquema de funcionamiento:

1.- El demandante de recursos, cliente, solicita información al servidor.

2.- La petición del cliente es recibida por parte del servidor.

3.- Dicha solicitud es procesada por el servidor.

4.- El servidor envía el resultado al demandante del servicio.

5.- El cliente procesa el resultado al recibirlo.

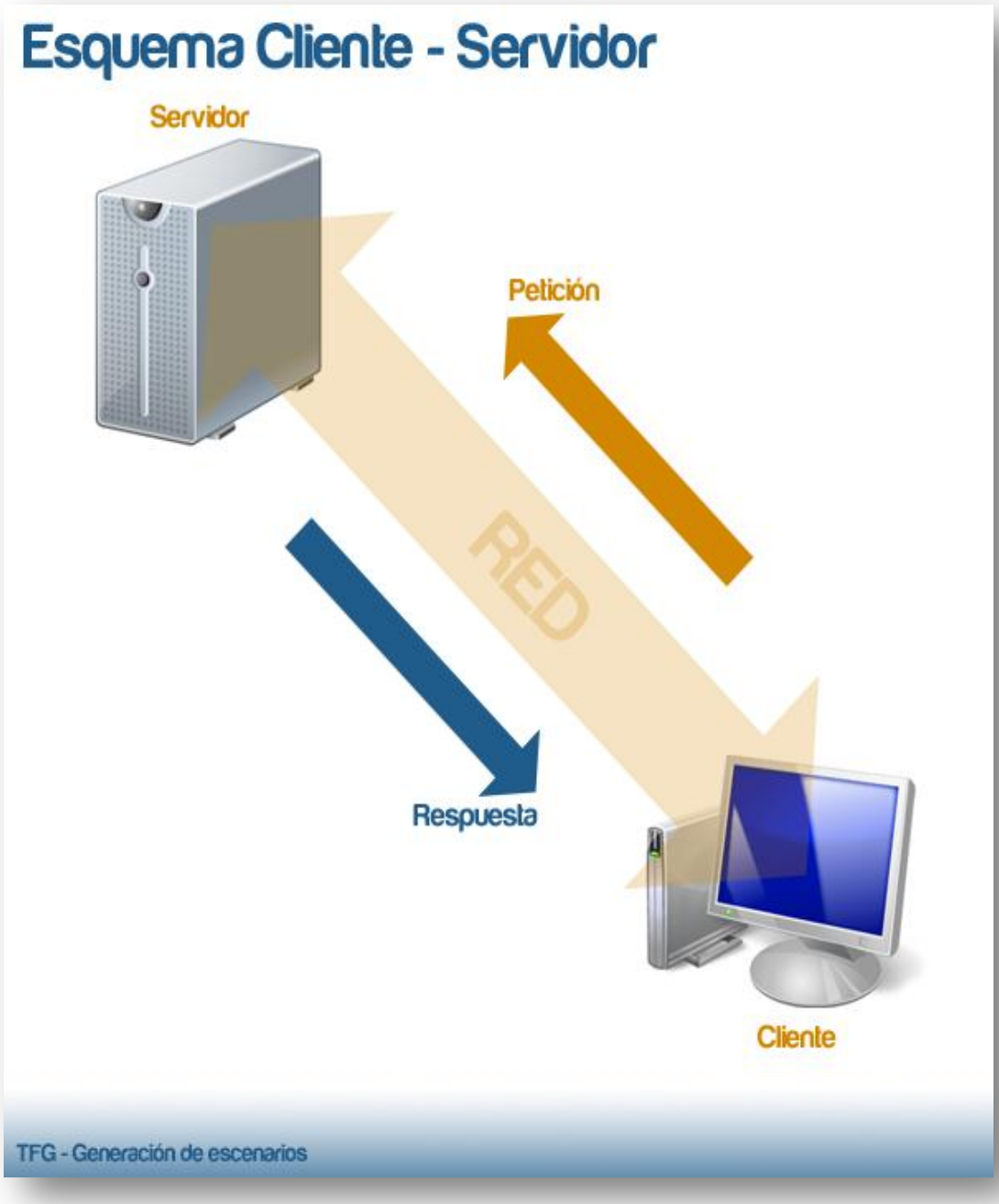


Ilustración 24. Esquema cliente - servidor

Mediante la utilización del modelo basado en esta arquitectura se puede conseguir un sistema cuyo control es totalmente centralizado, puesto que los recursos y accesos son controlados por el servidor, convirtiendo al sistema más seguro en cuanto a defectos en programas provenientes por parte del cliente y facilitando tareas de actualización u otros recursos. Presenta, además, muchas opciones de facilidad en el aspecto del mantenimiento e integración de nuevas tecnologías, favoreciendo la escalabilidad de las soluciones. [36]

Según esta estructura, el papel tomado por el cliente, llevará a cabo las siguientes funciones, entre otras:

- Administración de la interfaz de la aplicación.
- Procesamiento de la lógica de la aplicación.
- Realización de validaciones locales.
- Interacción con el usuario de la aplicación.
- Generar solicitudes de bases de datos.
- Recibir respuesta del servidor y formatear los resultados.

Mientras que el papel que toman los componentes del servidor, generalmente son funciones que poseen relación con las reglas de la capa de negocio y los recursos de datos:

- Aceptación de peticiones de bases de datos que se realizan por parte de clientes.
- Procesamiento de dichas peticiones o requerimientos.
- Formateo de la información para su correcta transmisión al cliente.
- Procesamiento de la lógica de la aplicación.
- Validación a nivel de bases de datos.

En el siguiente diagrama de componentes se muestra cada módulo de la aplicación basándonos en el modelo anterior, detallando las operaciones que realiza cada uno de ellos.

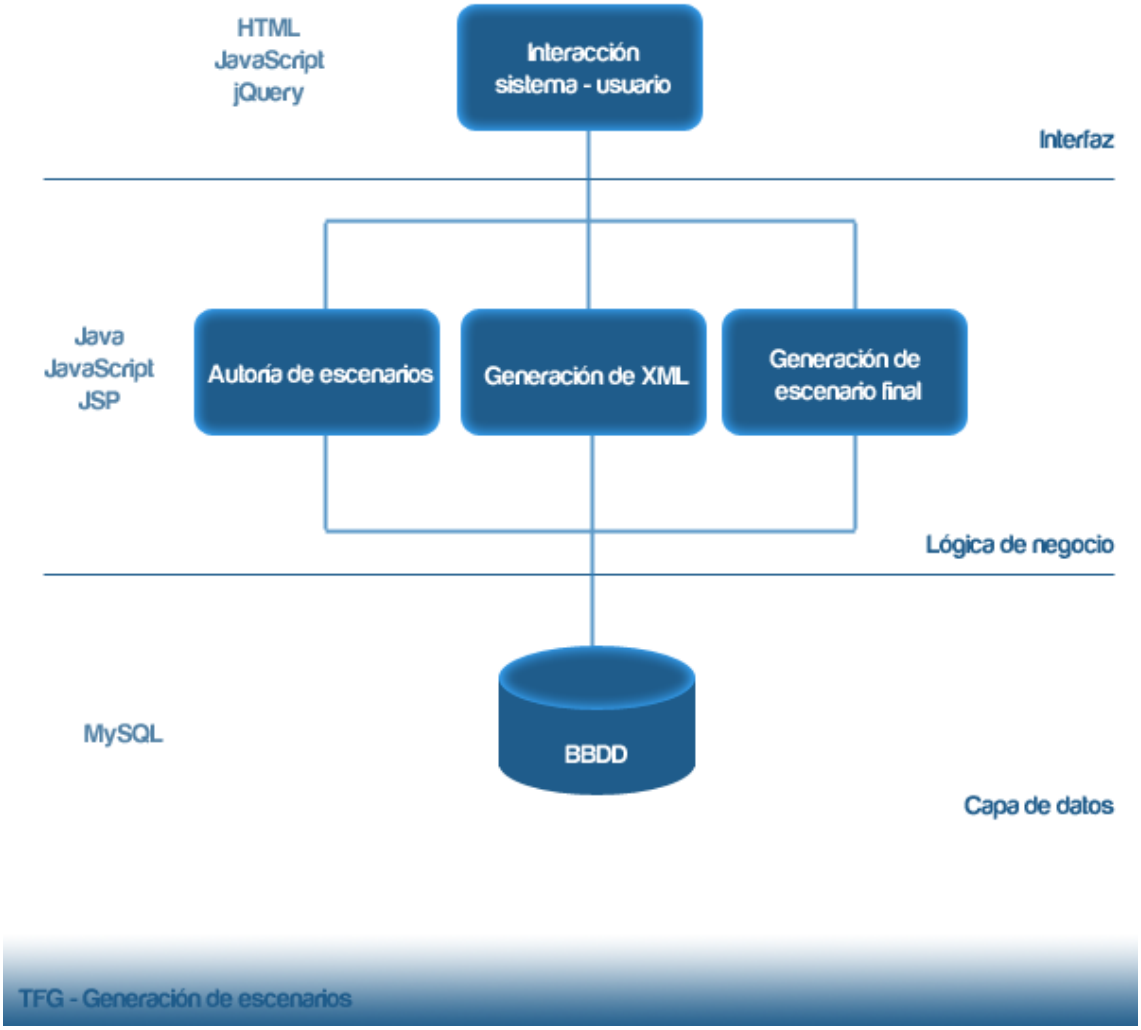


Ilustración 25. Diagrama de componentes

En las siguientes secciones se desarrollará detalladamente cada capa anteriormente descrita en el diagrama de componentes de la ilustración _ y cada módulo que las componen.

LÓGICA DE NEGOCIO

En cuanto a la lógica de negocio, está compuesta por el módulo central de la aplicación, encargado de la ejecución del sistema. Realiza el control de todas las fases del proceso de creación del escenario. Desde este módulo se accede a la base de datos, que se encuentra en la capa de datos, tal y como queda representado en el

diagrama anterior, con el fin de recuperar la información necesaria de la misma, como por ejemplo, los datos que poseen escenarios ya creados, a los que se les quiere añadir una nueva escena. Este módulo utiliza los conceptos que quedan reflejados en la lógica de la aplicación, donde se controlan múltiples aspectos de la herramienta en sí.

Entre ellos se encuentra todo el control relacionado con la conexión y acceso a la base de datos, mencionado anteriormente, y la utilización de dicha información para realizar un acceso a la lógica encargada de la generación del documento XML, que será utilizado en la aplicación externa, y que contiene los datos que han sido interpretados a lo largo del proceso de creación del escenario.

En este módulo, que actúa como controlador, también se encuentran todos los conceptos encargados de controlar y establecer el comportamiento de cada una de las fases del proceso y su correcto funcionamiento.

A continuación, se procederá a desglosar esta capa de negocios en los siguientes tres módulos: Autoría de escenarios, Generación de documento XML y Generación del escenario final.

AUTORÍA DE ESCENARIOS

Este módulo es el encargado de controlar el funcionamiento de todos los elementos relacionados con la creación de las escenas que se aplicarán a los escenarios, con el fin de ser posteriormente generados.

La creación y modificación de escenarios se tratan de la misma manera mediante este módulo. Es el encargado de todas las acciones relativas a la creación del nuevo escenario o la selección de uno ya existente, para ello debe obtener información almacenada sobre los mismos.

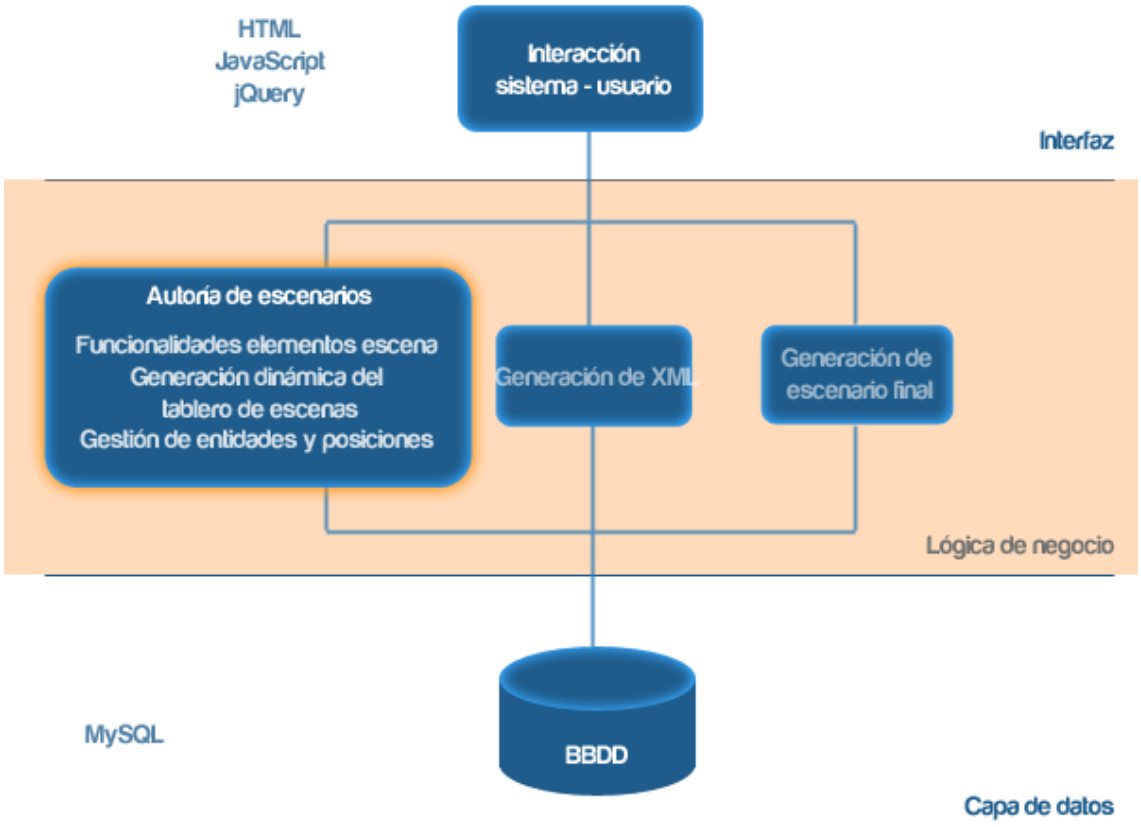


Ilustración 26. Diagrama de componentes. Autoría de escenarios

Otra funcionalidad de la lógica de la aplicación es la correcta implementación del algoritmo que genera el tablero de casillas que compone el escenario, para ello también realiza una conexión a la capa de datos para obtener la información del tamaño introducida en fases previas por el usuario. Dependiendo de estos valores generará de forma dinámica un tablero con distinto número de celdas, representando tanto paredes como suelos. Todo esto se detallará en el capítulo dedicado a la implementación de la herramienta en este documento.

Además, gestiona todas las funcionalidades de determinación de tamaños de escenas y la composición de las mismas. Esto comprende el sistema de establecimiento de texturas, tanto en paredes como en suelos de la escena; la funcionalidad de determinación de las alturas en las escenas y el posicionamiento de las entidades en las mismas.

GENERACIÓN DEL XML

Este es el módulo encargado de realizar la generación del documento XML que servirá como fuente de datos a la aplicación que lo utiliza para generar el escenario en tres dimensiones mediante la herramienta Unity 3D. Utiliza todos los datos recopilados durante el proceso de creación (controlado por la lógica del módulo de autoría de escenarios) y genera un fichero a través de un *parser* con dicha información. Este

módulo, al igual que el anterior, se ayuda de la capa de datos mediante accesos a la base de datos para realizar correctamente su funcionalidad.

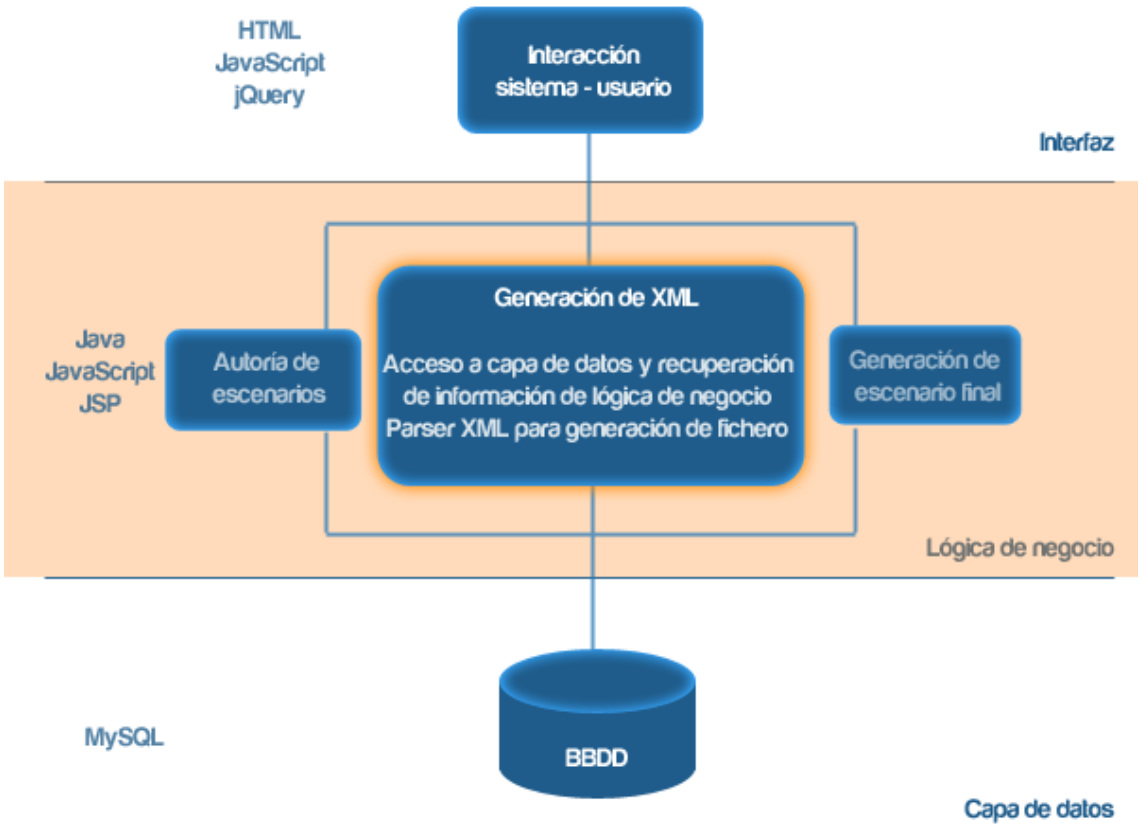


Ilustración 27. Diagrama de componentes. Generación de XML

GENERACIÓN DEL ESCENARIO FINAL

Este módulo es el que controla la relación entre el resultado obtenido por esta aplicación de creación de escenarios para juegos educativos, con la encargada de comunicarse con la herramienta Unity3D para generar dicho escenario basándose en el contenido generado por el anterior módulo.

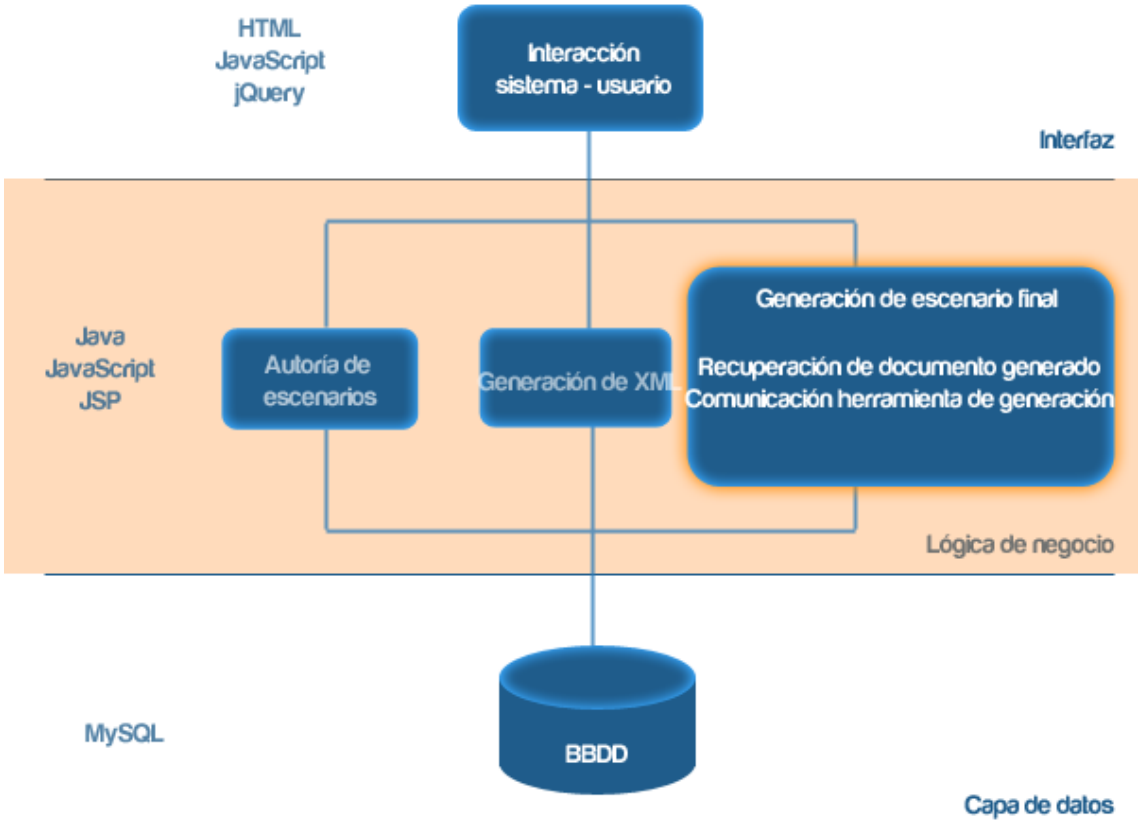


Ilustración 28. Diagrama de componentes. Generación de escenario final

CAPA DE DATOS

Esta está compuesta por las tablas creadas en la base de datos, que contienen la información de los escenarios que han sido creados, así como las escenas que componen a cada uno de ellos. Desde la lógica de negocios se recupera toda esta información para el correcto funcionamiento de la aplicación y por medio de accesos entre ambas capas se obtienen datos que serán determinantes para el comportamiento de la herramienta en diversas fases del proceso de creación de escenarios.

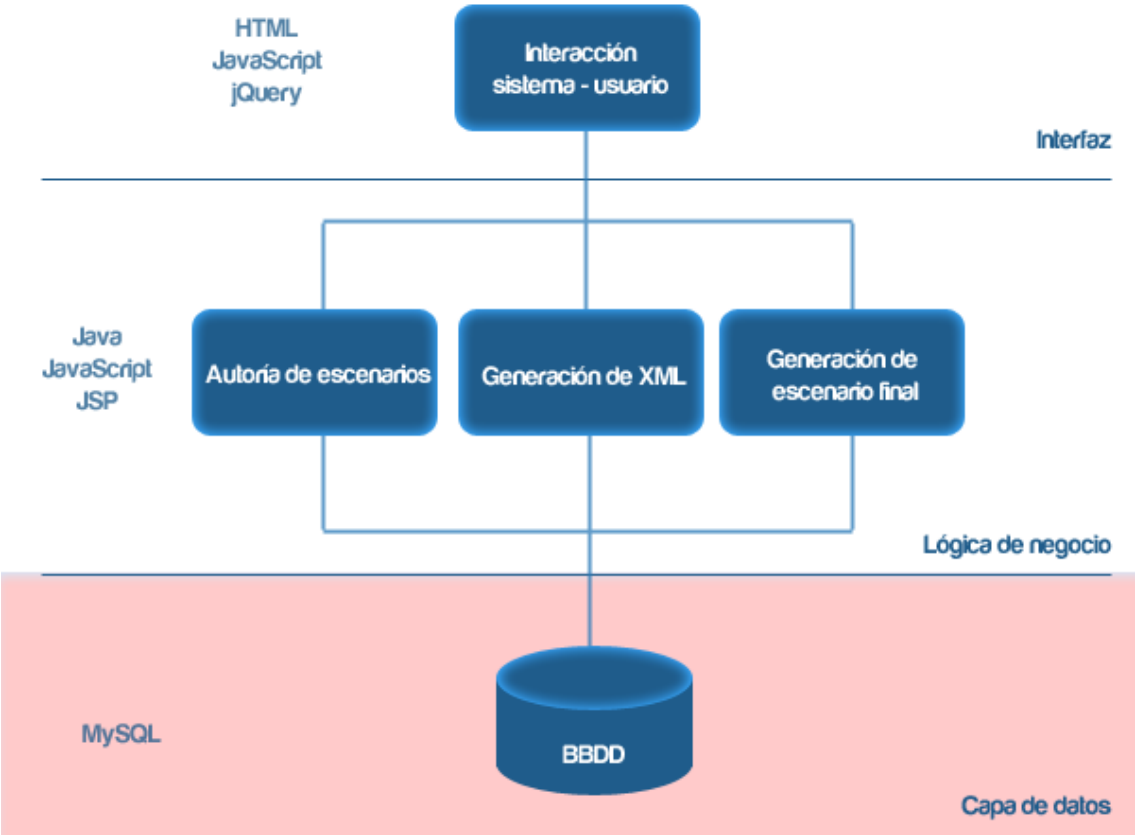


Ilustración 29. Diagrama de componentes. Capa de datos

A continuación se presenta la definición del diseño de la Base de Datos de esta aplicación, mostrando el diagrama Entidad-Relación de la misma.

4.2.1 DIAGRAMA DE SECUENCIA

A continuación se muestra un diagrama de secuencia de la ejecución completa de las funcionalidades de la aplicación, en la que aparece la interacción entre el sistema y el usuario, mostrando la arquitectura basada en Cliente – Servidor. En ella se representa la ejecución y las funcionalidades de la creación simple de una escena en un escenario determinado, hasta la generación del documento XML.

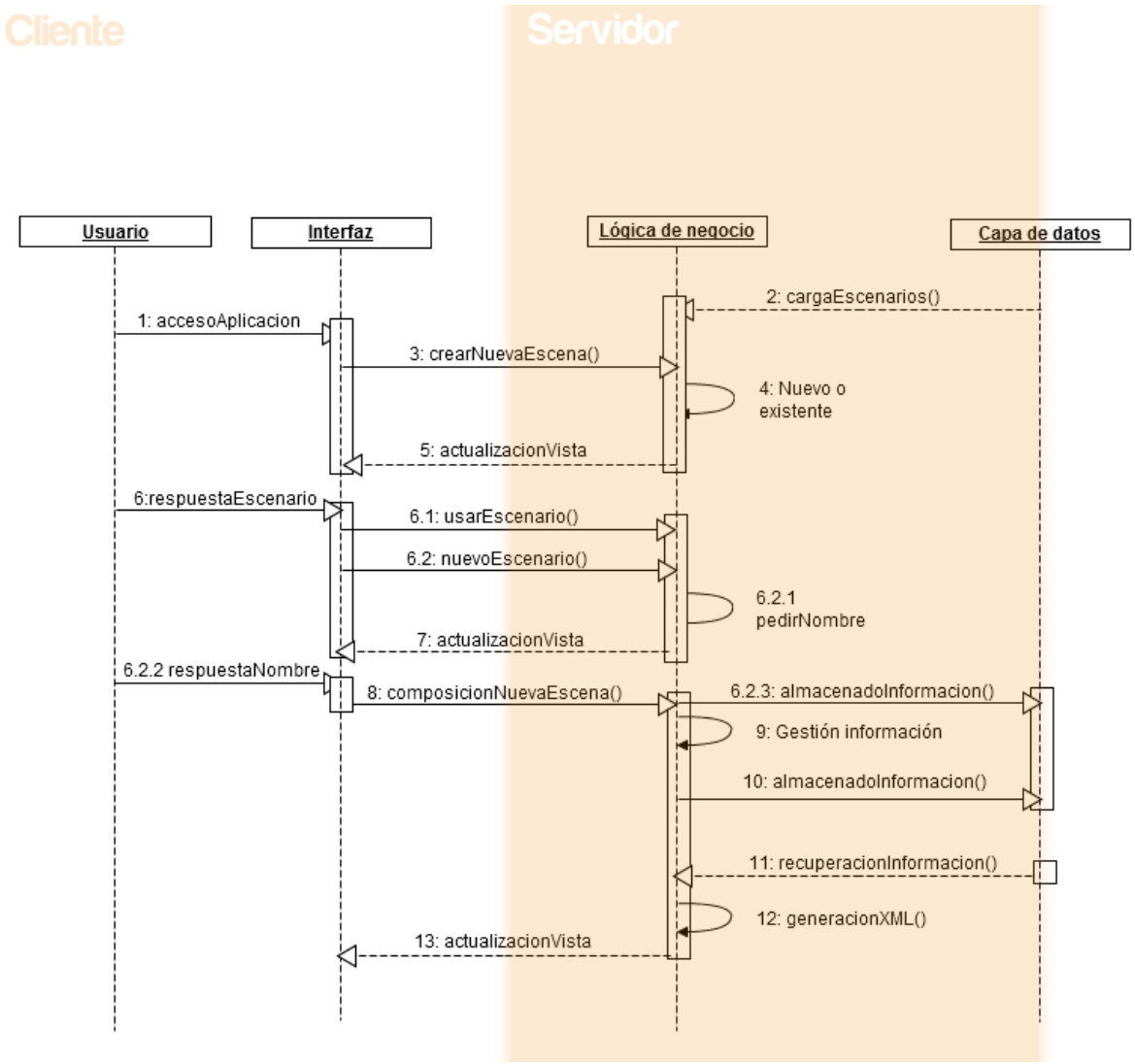


Ilustración 30. Diagrama de secuencia de los componentes de la aplicación

En el diagrama de secuencia anterior (Figura 30) se observan las funcionalidades de la aplicación sucedidas entre componentes. A continuación se procede a su explicación:

1. Acceso a la aplicación, por parte del usuario, que visualiza la interfaz de la aplicación.

2. Se procede a la carga de los escenarios que se han generado mediante la aplicación, con el fin de que la lógica de negocio los procese y se muestren en la herramienta, preparados para la siguiente acción.

3. Se inicial el proceso de creación de la nueva escena, es aquí donde también se hace uso de la información previamente cargada de los escenarios existentes, descrita en el anterior paso 2.

4. Se da opción al usuario de escoger entre uno de estos escenarios, y a comenzar a trabajar sobre todo lo procesado por la lógica de negocios en los pasos 2 y 3. Esto se hace visible al usuario mediante el siguiente paso.

5. Se actualiza la vista de la interfaz para el usuario, donde procederá a escoger entre un usuario nuevo u otro ya creado.

6. Respuesta del usuario ante la elección de selección de escenario.
    a. Uso de un escenario ya existente. (6.1)
        i. Se actualiza la vista. (Paso 7)
        ii. El usuario interactúa con la interfaz, seleccionando el tamaño de la escena.

    b. Creación de un escenario nuevo. (6.2)
        i. La aplicación necesita un nombre identificador del nuevo escenario. (6.2.1)
        ii. Se actualiza la vista con el fin de que el usuario introduzca el nombre de dicho escenario. (Paso 7)
        iii. El usuario interactúa con la interfaz, seleccionando nombre y tamaño de la escena. (6.2.2)
        iv. El identificador del nuevo escenario se almacena en la capa de datos. (6.2.3)

7. Actualización de la vista de la aplicación.

8. Se inicia el proceso de composición de la nueva escena con los datos introducidos.

9. La lógica de negocio procesa todo lo configurado en esta fase del proceso.

10. Se almacenan todos los datos procesados referentes a la composición de la escena.

11. Se recupera la información necesaria para el inicio del paso siguiente.

12. Se procede a generar el XML de la escena generada.

13. Se actualiza la vista para que el usuario finalice el proceso.

DIAGRAMA ENTIDAD-RELACIÓN

Mediante este diagrama entidad-relación se puede representar todas las entidades que presentan relevancia del sistema, junto con sus propiedades e interrelaciones. Sobre este modelo destacan los siguientes elementos:

- Entidad: se trata del objeto sobre el que se desea almacenar la información y está compuesto de atributos, representando los datos que completan la definición del objeto. Este puede poseer existencia de carácter físico o conceptual, distinguiendo así entre entidad concreta y entidad abstracta.

  Entre los atributos que la componen siempre habrá uno de ellos, o un conjunto que no va a ser repetido, denominando esto como clave de la entidad.

  La representación de la entidad, sus atributos y la clave en el diagrama entidad-relación será de la siguiente forma:

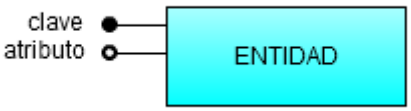


Ilustración 31. Representación de entidad en diagrama entidad-relación

- Relación: es la asociación entre las entidades, describiendo la dependencia entre las mismas.

  En el diagrama queda representado mediante un rombo, con una etiqueta descriptiva de la relación.

Ilustración 32. Representación de relación en diagrama entidad-relación

Estas relaciones presentan cardinalidad entre las entidades. La nomenclatura utilizada en esta ocasión es: *(1,1) (1, N) (0,1) (0, N) y (N, M).*

A continuación se muestra el diagrama entidad-relación basado en el sistema de información:

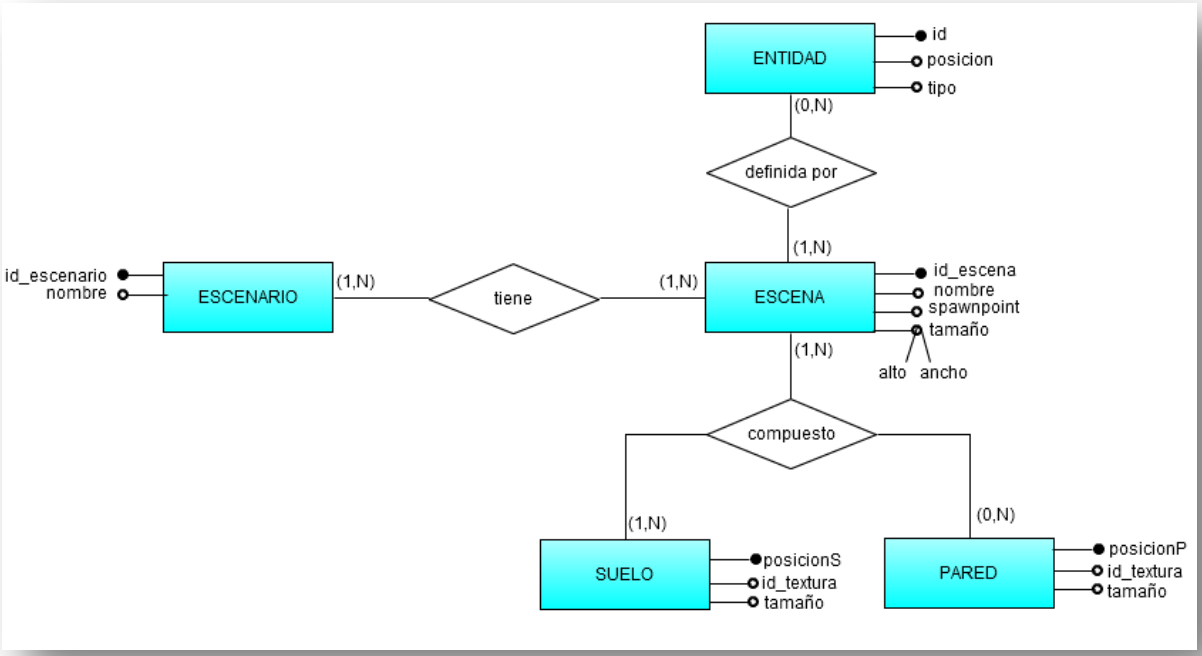


Ilustración 33. Diagrama entidad-relación

En el diagrama anterior podemos observar las cinco entidades relevantes que componen la totalidad del sistema. Se tratan de Escenario, Escena, Entidad, Suelo y Pared.

Se puede observar como es el Escenario la entidad 'principal', pues es la que está compuesta de las Escenas, y esta última, a su vez, queda definida por las Entidades, Suelos y Paredes. Por lo tanto es posible afirmar que la entidad que representa las Escenas es la más relevante, aun no tratándose de la entidad principal.

La entidad Escenario posee como atributos su número identificador y el nombre que lo define, siendo el primero clave de la entidad, al igual que la entidad de mayor relevancia, que representa la Escena, y que también posee como atributos su clave de identificación y su nombre, a los que se suman el tamaño de la escena (compuesto por alto y ancho) y el punto de reaparición del personaje.

Como se puede observar, la escena queda definida a través de Entidades, cuyos atributos son la posición y el tipo de la misma, y como clave de la entidad, su número identificativo.

Por último, la escena queda compuesta por dos entidades más, las que representan el Suelo y la Pared. Ambas poseen atributos similares: números de identificación de texturas, tamaño y posición de cada uno de ellos, siendo este último atributo la clave en las respectivas entidades.

INTERFAZ, CAPA DE PRESENTACIÓN

Esta es la encargada de ofrecer por pantalla toda la funcionalidad de la aplicación, con el objetivo de que el usuario interactúe con la misma y logre satisfacer todas las necesidades.

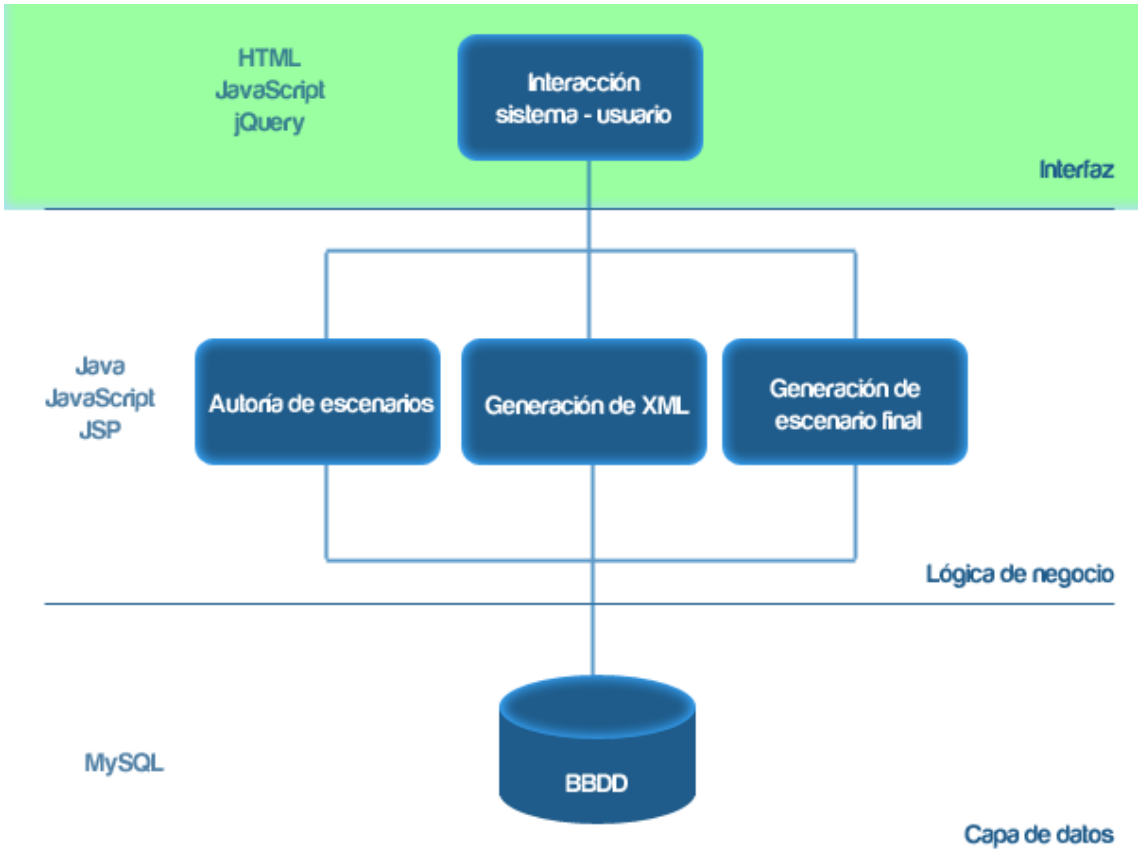


Ilustración 34. Diagrama de componentes. Interfaz

4.3 DEFINICIÓN DEL ASPECTO DE LA APLICACIÓN

En este apartado se realiza un boceto del aspecto de la interfaz gráfica que deberá presentar la aplicación web. Esta es una de las fases más importantes, puesto que es la parte de la aplicación donde va a actuar el usuario, y donde va a interaccionar con el fin de obtener solución a sus necesidades. Debe tratarse, por lo tanto, de una interfaz intuitiva y no excesivamente compleja.

A continuación se muestra el diseño inicial, después de realizar el estudio del análisis del sistema, surgido durante las primeras reuniones con el cliente. Esto es necesario para posteriormente tener una base sobre la que empezar a desarrollar la

aplicación. Por supuesto, el resultado final del producto puede sufrir modificaciones tras el proceso de implementación, pero estos bocetos facilitan el inicio del desarrollo del sistema y sirven como declaración de cómo puede responder la aplicación ante las acciones del usuario, cómo se pretende que funcione el sistema de navegación de las pantallas de la herramienta y los diversos elementos de control del sistema.

Además, también ayuda al cliente a realizar una toma de contacto con la herramienta y poder realizar evaluaciones iniciales, mostrando o no su conformidad al equipo de desarrollo.

En primer lugar, se muestra la idea de selección del tamaño del escenario y de la escena.

4.3.1 PANTALLA DE SELECCIÓN DEL TAMAÑO DEL ESCENARIO

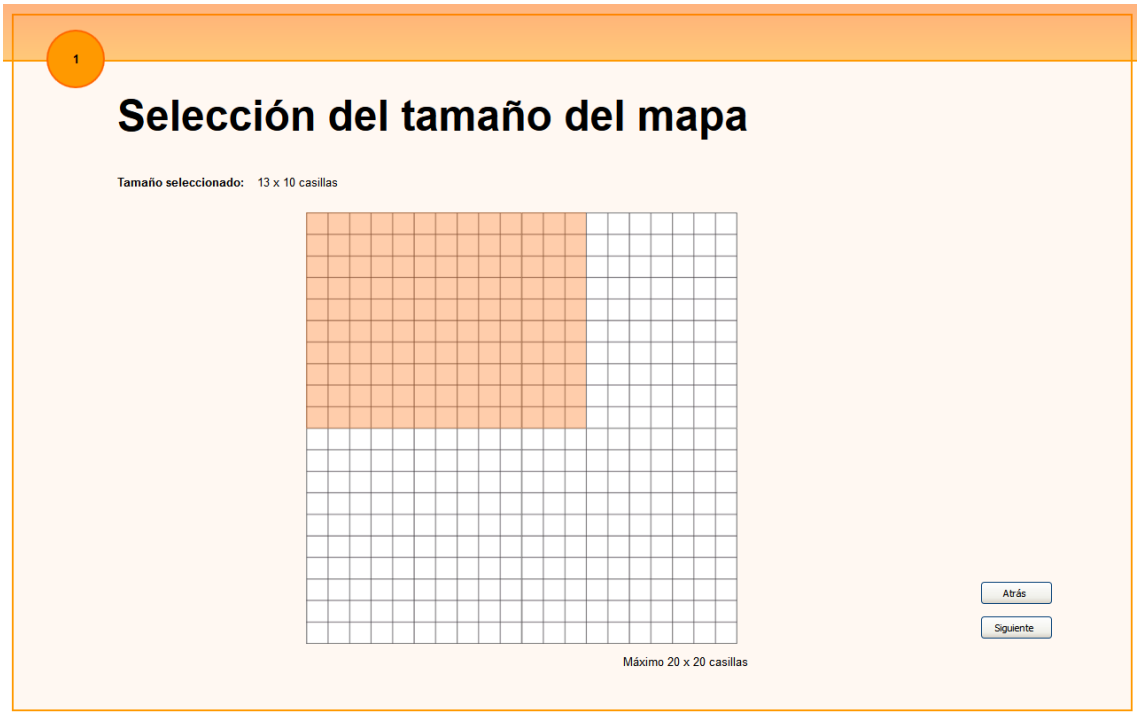


Ilustración 35. Pantalla de selección del tamaño del escenario del prototipo

En esta primera pantalla (Ilustración _) se observa cómo se permite al usuario la opción de seleccionar el tamaño del escenario. Esto se realizaría con un solo clic en la casilla hasta la que se quiere determinar el ancho y alto deseados, como queda representado en las siguientes ilustraciones (Ilustración _ e Ilustración _):

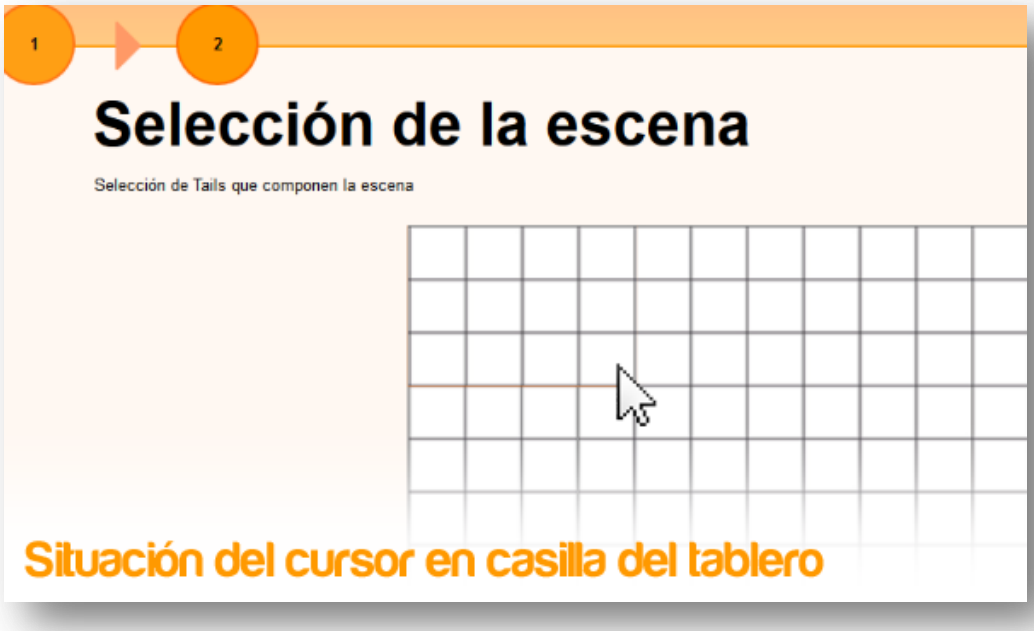


Ilustración 36. Detalle de selección de tamaño con cursor. Prototipo

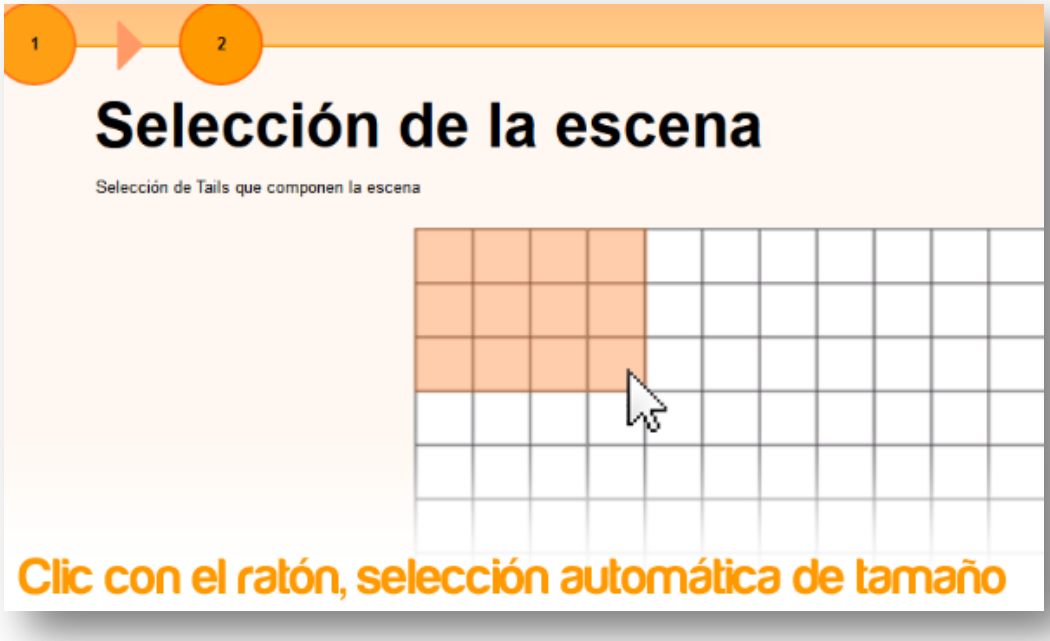


Ilustración 37. Detalle de selección de tamaño con cursor II. Prototipo

Los casos de uso que fueron analizados en la etapa anterior del estudio del proyecto que son tratados en esta parte de la interfaz, son los siguientes:

CU E2-04 - **Composición en el escenario**, caso de uso obtenido según los requisitos relacionados con la elección del tamaño del escenario y de escenas, que en reuniones posteriores con el cliente fue modificado. La idea inicial era seleccionar el tamaño del escenario en la primera fase del proceso, pero más tarde se apreció más

óptimo que el propio escenario, tanto su forma como el tamaño, fuera definido una vez se iban añadiendo las escenas.

Esta fase es la se basa en el caso de uso **CU E1-01 – Crear un nuevo escenario**, en el caso de que se inicie dicho escenario en ese momento, puesto que para el proceso de creación de una nueva escena también es posible seleccionar uno ya existente (**CU E1-02 – Seleccionar escenario existente**). Esto se decidió en reuniones posteriores a la creación del prototipo inicial tras decisiones por parte del cliente y el desarrollador, llevando a cabo modificaciones en los casos de uso y en los requisitos.

4.3.2 PANTALLA DE SELECCIÓN DEL TAMAÑO DE LA ESCENA

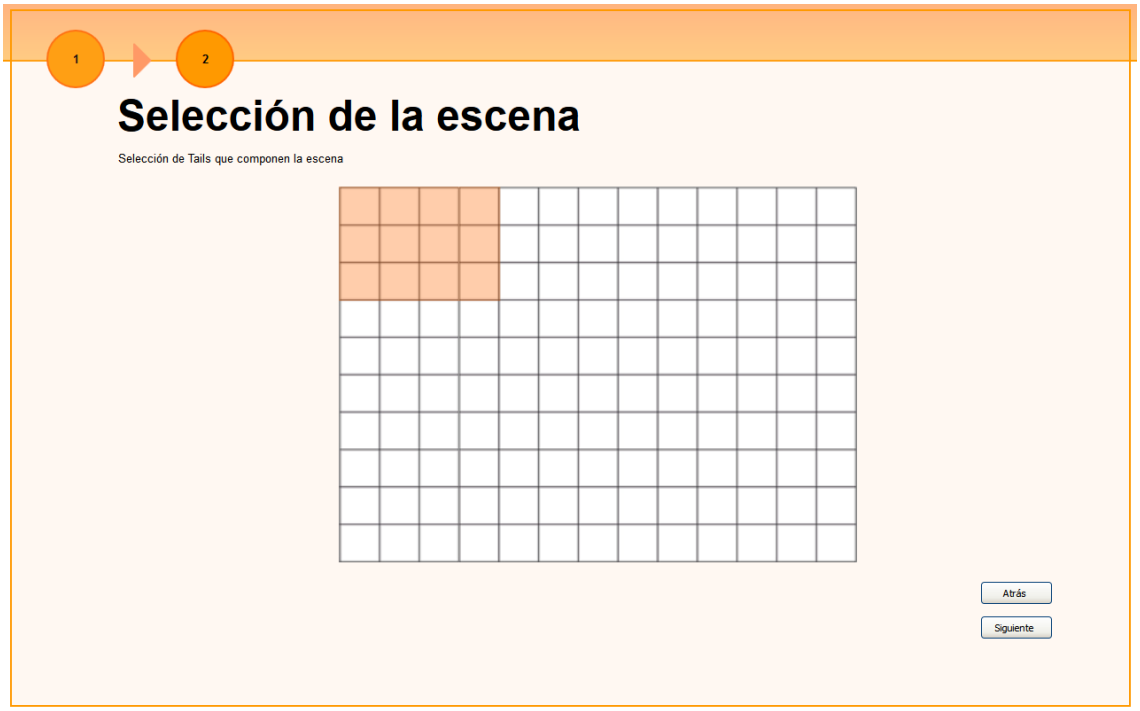


Ilustración 38. Pantalla de selección de tamaño de escena del prototipo

En esta ocasión se utiliza el mismo método de selección de tamaño, pero en esta ocasión queda determinado para la escena que compondrá parte del escenario.

Al igual que lo explicado anteriormente, el usuario tendrá que seleccionar mediante el cursor las casillas que representan tanto el ancho como el alto de la escena. La funcionalidad de esta pantalla es similar a la explicada en el apartado anterior, mediante las ilustraciones _ y _.

Los casos de uso relacionados con estas pantallas del prototipo son los siguientes:

CU E2-02 – Selección del tamaño de la escena. Al igual que anteriormente, este caso de uso se basa en requisitos iniciales que posteriormente fueron modificados. Una de las modificaciones que se realizaron en dichos requisitos por parte del cliente y el desarrollador, fue la de añadir en esta fase la determinación del nombre que identificará a la escena en cuestión. Es por esto, que también se debe tratar el caso de uso **CU E2-01 – Introducción del nombre de la escena**, a través de un campo de entrada que permite al usuario completar dicha acción. En el prototipo inicial vemos esta funcionalidad en la siguiente pantalla.

La idea del sistema de navegación entre las distintas pantallas de la aplicación aparece presente en los anteriores prototipos, tras completar cada fase del proceso de creación del escenario se utilizarían los botones 'Atrás' y 'Siguiente' para avanzar o volver atrás en cada una de las etapas. Se puede observar esta forma de confirmación o cancelación en todas las fases del proceso, mediante los botones representados en la Ilustración 39:

Ilustración 39. Detalle del sistema de avance y vuelta atrás del prototipo

Además, se puede observar que es posible conocer en que fase nos encontramos del proceso en todo momento, gracias a la zona superior de la herramienta, donde se muestra la numeración de la fase y su nombre, permitiendo también continuar navegando por cada una de ellas previamente completadas. Esto queda representado en la siguiente ilustración, Ilustración 40:

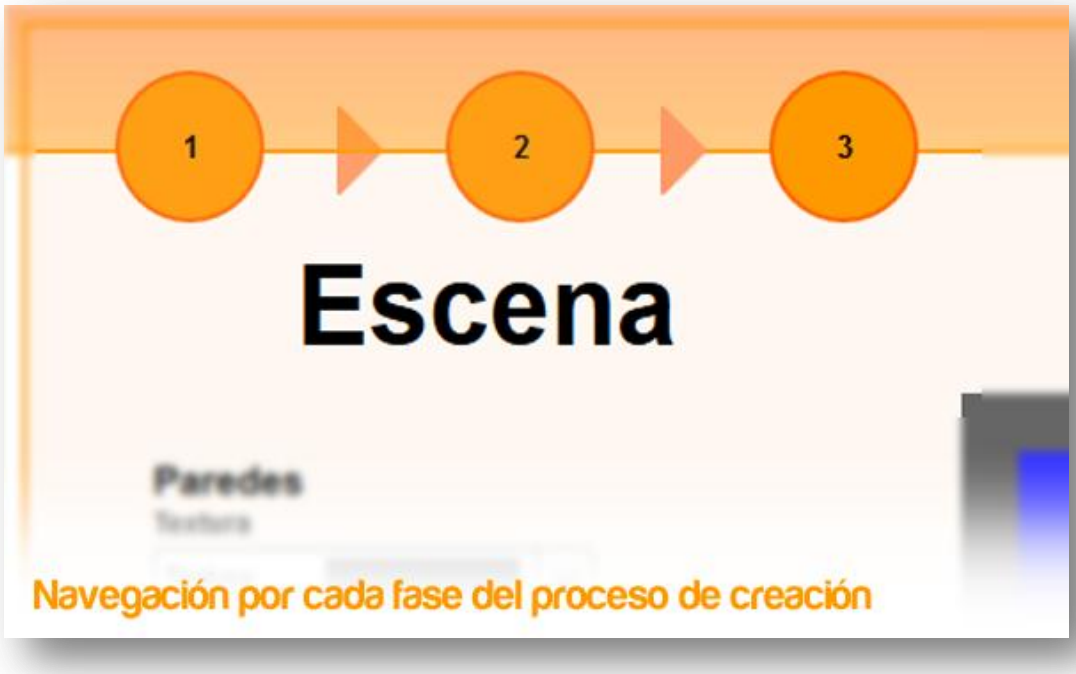


Ilustración 40. Detalle del sistema de navegación de la aplicación en el prototipo

4.3.3 PANTALLA DE COMPOSICIÓN DE LA ESCENA

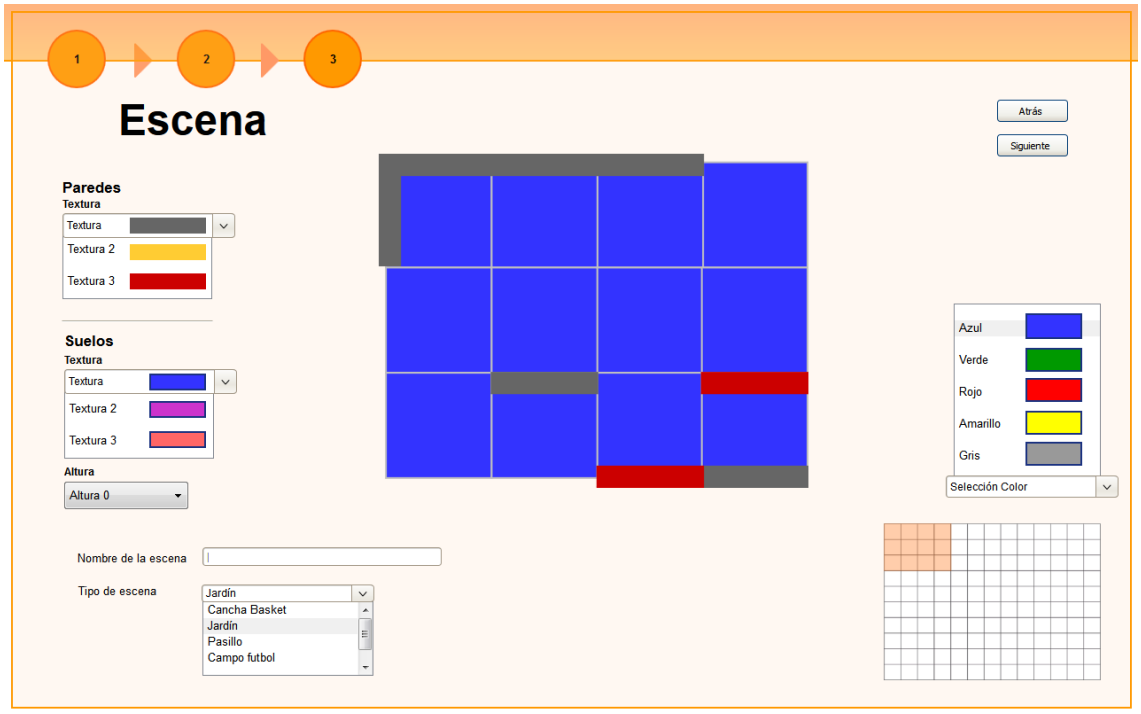


Ilustración 41. Pantalla de composición de escena del prototipo

En el boceto anterior (Ilustración 41) se puede examinar la idea de los elementos de control encargados de la etapa de composición de la escena. Tanto para las texturas de la pared y suelos, como la altura de estos últimos, el sistema de control

utilizado sería seleccionar la opción deseada en cada desplegable y aplicarlo sobre la vista de la escena, pudiendo observar las modificaciones en la misma.

Este proceso a realizar en la etapa de composición de la escena, queda representado en las siguientes ilustraciones:

CREACIÓN DE PAREDES

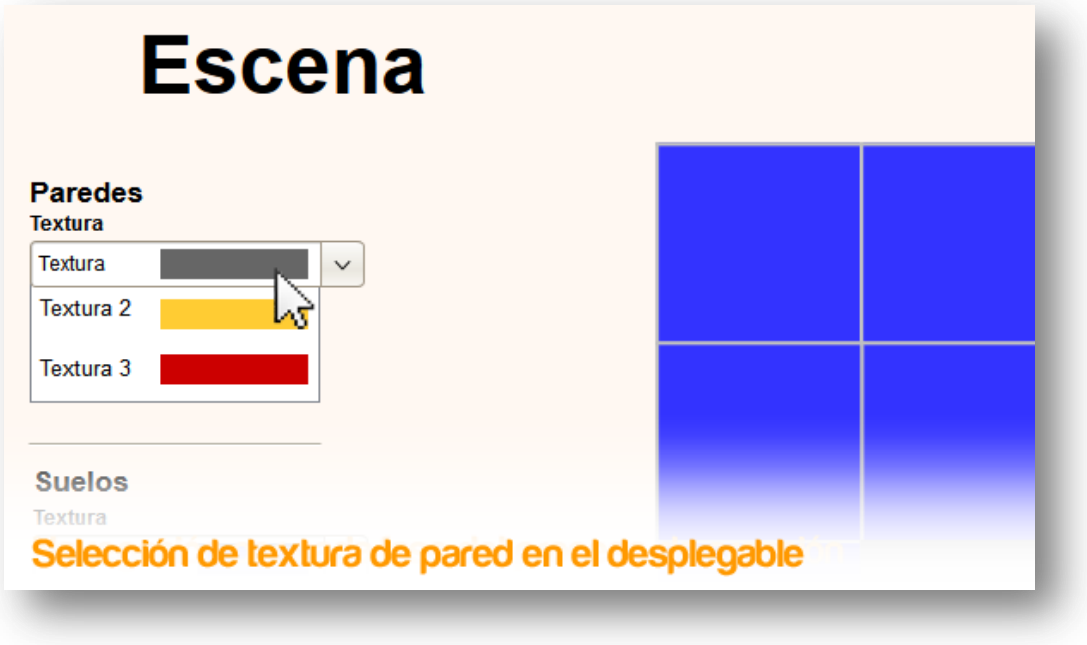


Ilustración 42. Detalle de creación de paredes en el prototipo

Ilustración 43. Detalle de creación de paredes en el prototipo II

Ilustración 44. Detalle de selección de texturas del suelo en el prototipo

Ilustración 45. Detalle de selección de texturas en suelo del prototipo II

Los casos de uso relacionados con estos apartados del prototipo inicial son los siguientes:

CU E2-03 – Edición de la escena, donde se determinan las propiedades de la misma una vez definido por parte del usuario el tamaño.

4.3.4 PANTALLA DE ESTABLECIMIENTO DE ENTIDADES

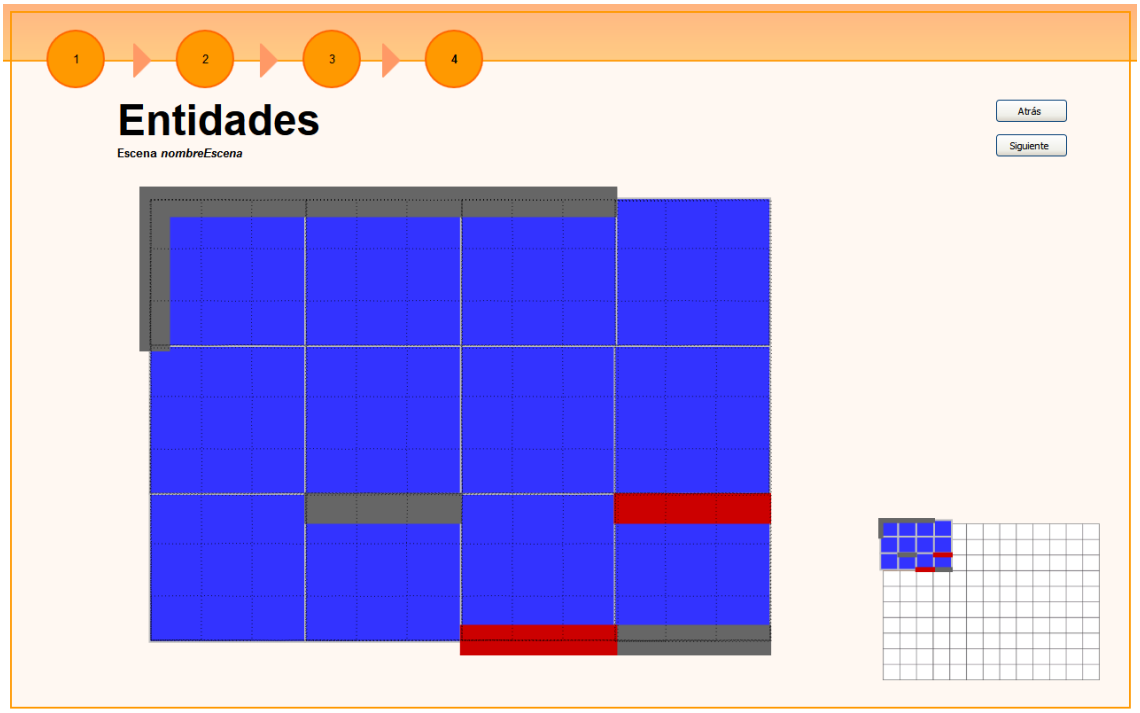


Ilustración 46. Pantalla de selección de entidades del prototipo

El sistema de control de las entidades según el boceto es muy similar, se aplicaría la entidad en la zona que se desee de la escena en cuestión, seleccionando previamente mediante un desplegable el tipo de entidad.

4.3.5 PANTALLA DE GUARDADO DE ESCENARIO

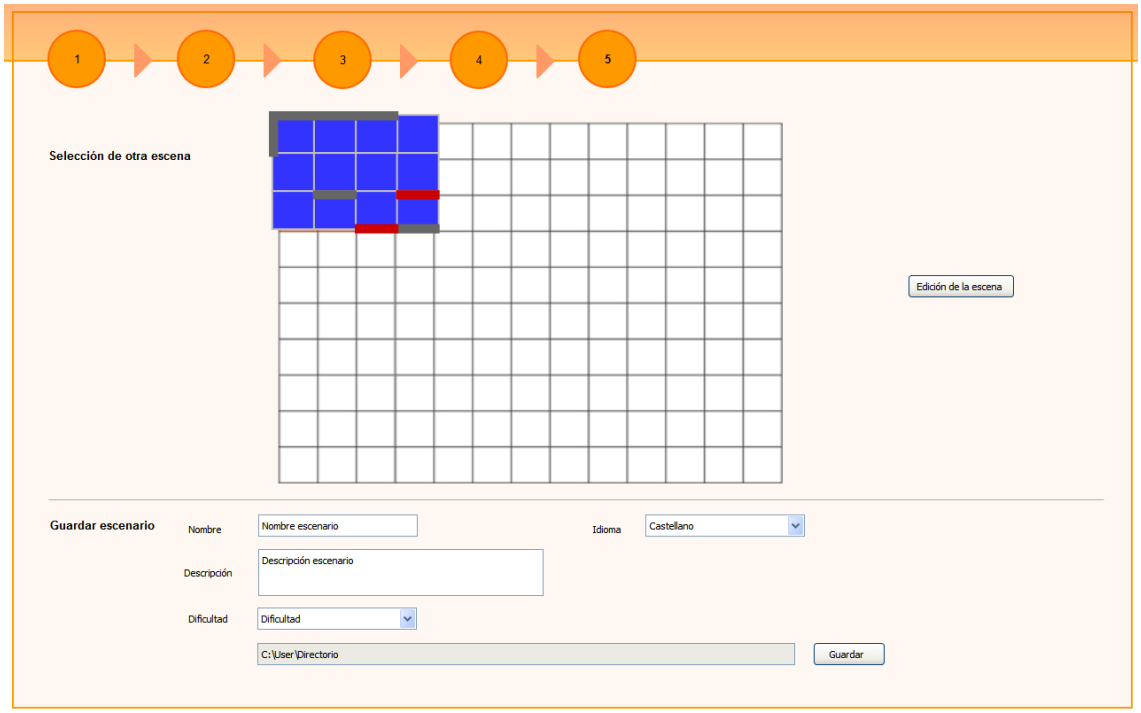


Ilustración 47. Pantalla de guardado de escenario en prototipo

4.3.6 PANTALLA DE CONFIRMACIÓN DE GUARDADO

Ilustración 48. Pantalla de confirmación de guardado del prototipo

Por último, se muestra el sistema de guardado del escenario, donde el usuario podría observar el resultado final y completar los datos del mismo a través de una serie de campos de entrada. En la siguiente pantalla, una vez completado el proceso, se da opción de abandonar la aplicación o continuar con la edición de una nueva escena para implementar en algún escenario. Remitiendo de esta forma al caso de uso **CU E1-03 – Salir de la aplicación**.

Basándose en este primer diseño o prototipo de la aplicación, se sientan las bases para su implementación real, modificando ciertos aspectos con el fin de hacer más eficiente e intuitiva la interfaz de la aplicación. Todas las modificaciones realizadas se podrán analizar en apartados siguientes en este documento, analizando el motivo de cada modificación y mostrando el resultado final.

5. IMPLEMENTACIÓN

En este capítulo de la documentación se describirá como se ha llevado a cabo la implementación de la arquitectura y el diseño del sistema, que ha sido detallado en los apartados anteriores. En esta ocasión, se mostrarán los aspectos más importantes de la fase de desarrollo tras el análisis y diseño del sistema, presentando las tecnologías utilizadas para la implementación del proyecto.

5.1 TECNOLOGÍAS UTILIZADAS

Tal como se avanzó en el capítulo del estado del arte, la aplicación será realizada mediante el lenguaje de programación Java, utilizando el entorno de desarrollo *Eclipse*. La herramienta puede ser ejecutada desde cualquier sistema operativo, mediante un navegador web.

Cabe destacar en el desarrollo de esta aplicación web el uso de *JavaScript* y de librerías como *jQuery*, [37] además de la técnica de desarrollo *Ajax*.

5.1.1 JAVA SERVER PAGES

En primera instancia, se procede a explicar la tecnología Java utilizada para generar contenido dinámico para la aplicación web, en forma de documentos HTML o XML, entre otros. Este conjunto de técnicas basadas en Java es denominado como *Java Server Page*s, también conocido por sus siglas, *JSP*.

Esta tecnología, permite la utilización de código Java mediante el uso de scripts, pudiendo ejecutar todo de forma similar a la que construye un *servlet*, pero de una manera más simplificada.

El sistema de funcionamiento de esta tecnología se basa en que el encargado de interpretar el código contenido en la página JSP, es el servidor web, en este caso Tomcat, que construye el código Java del servlet que se debe generar. Y es este servlet el encargado de formar el documento que se presentará al usuario en el navegador web.

A continuación, se muestra un ejemplo de página JSP de este proyecto, detallando las tecnologías presentes en el mismo.

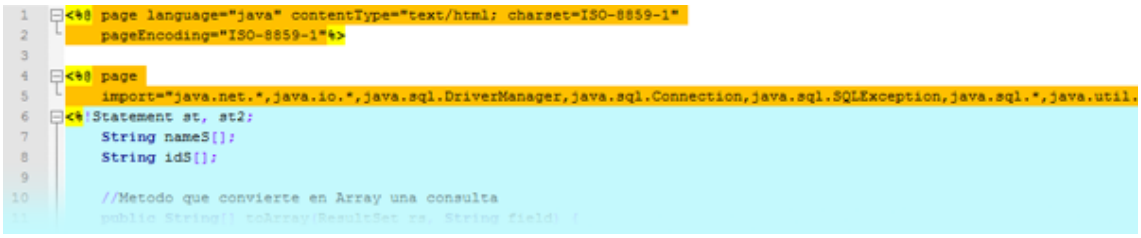


Ilustración 49. Fragmento de código. Ejemplo JSP

En la anterior ilustración (Ilustración 49) se muestra el inicio de una pagina JSP, la estructura de cabecera que tienen en común todas las páginas de este tipo.

Después se puede incluir la combinación de código Java denominados como *scriptlet*, usando la sintaxis normal *JSP*, *<%* y *%>*.

```
<%!Statement st, st2;
    String nameS[];
    String idS[];

    //Metodo que convierte en Array una consulta
    public String[] toArray(ResultSet rs, String field) {
        try {
            rs.beforeFirst();
            String str[];
            int rows = 0;

            while (rs.next()) {
                rows = rs.getRow();

            }

            str = new String[rows];
            int i = 0;
            rs.beforeFirst();

            while (rs.next()) {
                str[i] = rs.getString(field);
                System.out.println(field + "    " + str[i]);

                i++;
            }
            return str;
        } catch (Exception ex) {
            // handle the error
            System.out.println(" Salta excepcion ");
            return null;
        }
    }%>
```

Ilustración 50. Fragmento de código Java insertado en la página JSP

En esta ocasión, como se muestra en la ilustración 50, se puede observar el código java implementado en la página JSP. En este caso es interesante comentar el método, común a todas las demás páginas, utilizado para convertir todas las consultas que se realicen a la base de datos en *arrays*. Un método que hace muy cómodo el

tratamiento de la información contenida en la base de datos para trabajar con ella y mostrarla al usuario.

```
79  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/x
80  <html xmlns="http://www.w3.org/1999/xhtml">
81  <head>
82  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
83  <title>SG - Design Scene</title>
84  <meta name="description" content="Web Application" />
85  <meta name="keywords" content="web, application" />
86  <script type="text/javascript" src="script/jquery.js"></script>
87  <link href="css/main.css" rel="stylesheet" type="text/css" />
88
89  <script language="JavaScript" type="text/JavaScript">
90      function ratonOver(celda){
91          celda.style.backgroundColor="orange";
```

Ilustración 51. Fragmento de código HTML en página JSP

A continuación, se puede observar en la ilustración 51 el inicio de las etiquetas *HTML*, donde se presenta la importación de ficheros. En la línea 87, se puede observar la importación del fichero *.css*, la hoja de estilo que contiene la información y definición de estilos que serán aplicados en la generación de las páginas en el navegador.

También se puede ver la información contenida en las etiquetas meta, y la importación de la librería *jQuery*, en la línea 86 con el fichero *javascript .js* correspondiente.

5.1.2 LIBRERÍA JQUERY Y AJAX

Para el desarrollo de este proyecto, también se ha hecho uso de la biblioteca de JavaScript con el fin de manejar eventos, desplegar animaciones e, incluso, añadir interacciones mediante la técnica de AJAX en la aplicación web.

Por ello, también se procede a definir la técnica de desarrollo web para esta generación de interacción mediante AJAX, siendo esto ejecutado en el navegador del usuario. Observamos un ejemplo de ello en el código desarrollado para la aplicación web del proyecto.

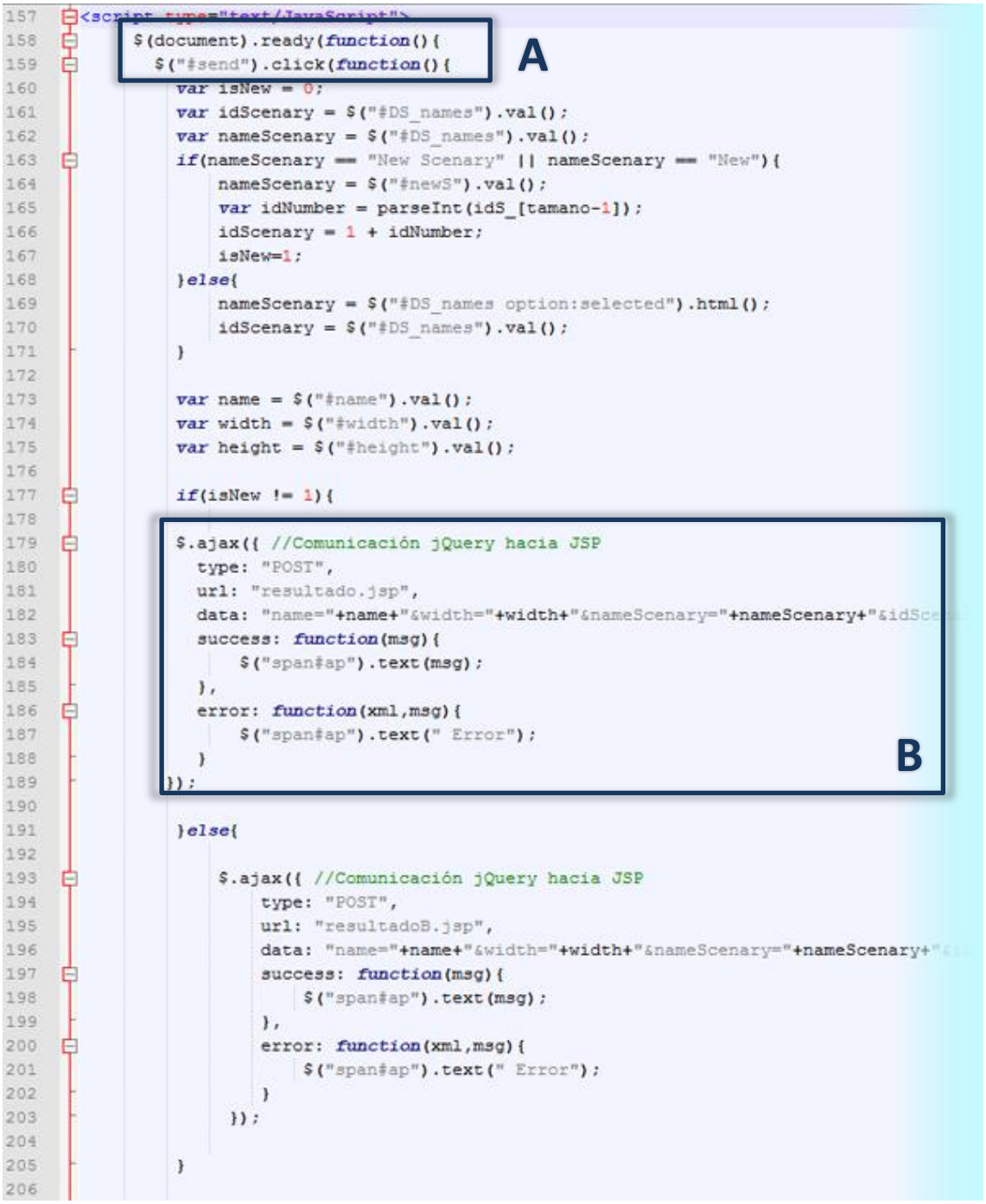


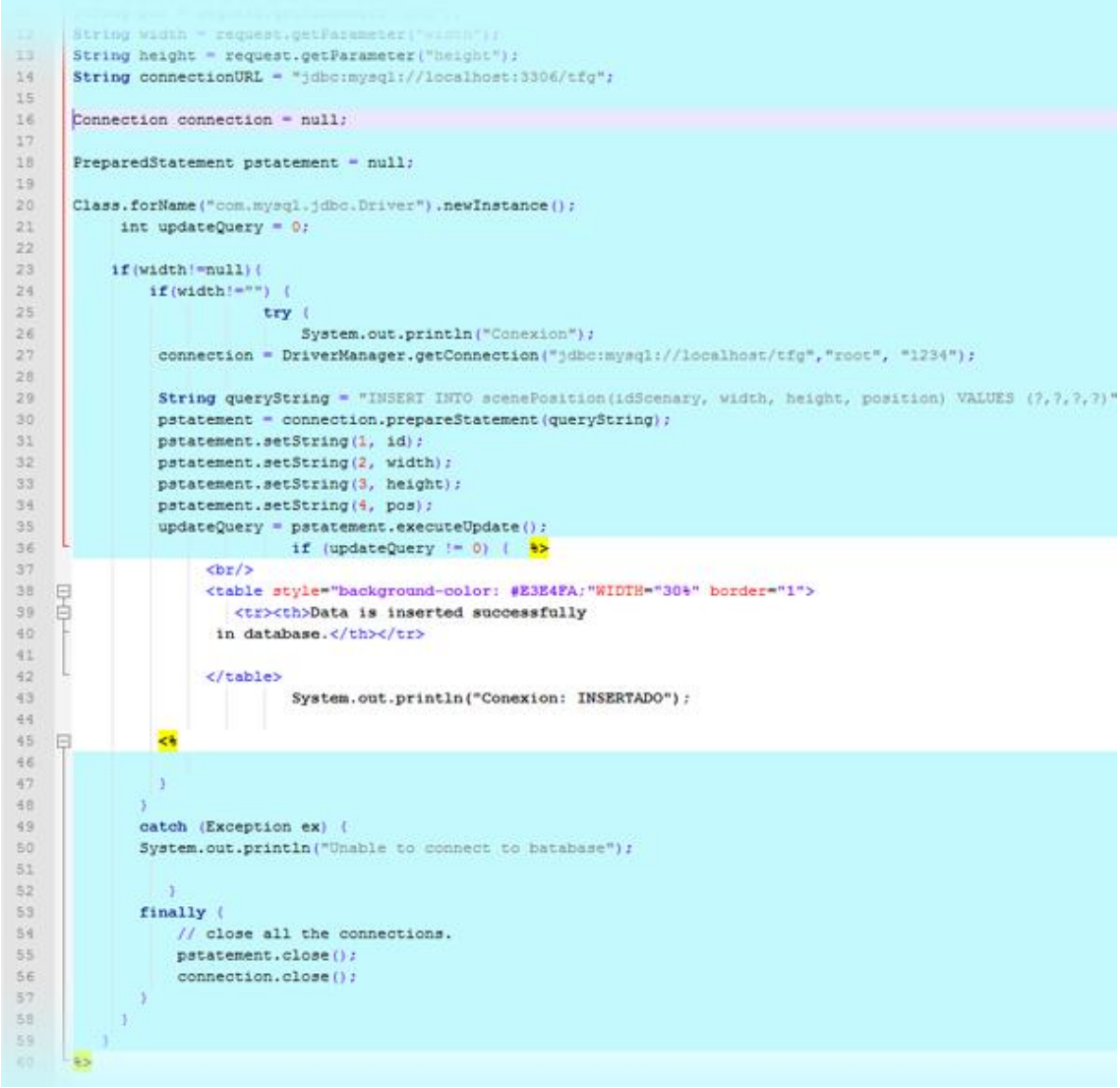
Ilustración 52. Fragmento de código con comunicación *jQuery* y *AJAX* en JSP

Podemos observar en la ilustración 52, cómo se hace uso de lo descrito anteriormente, utilizando *jQuery* (área A en la ilustración 52) y *Ajax* (área B en la ilustración 52) con el fin de enviar información a otra página *JSP*, gracias a esta comunicación utilizando la tecnología *jQuery*, mediante el uso de *Ajax*, hacia la página *JSP*.

Como se puede observar, en este caso se envían diferentes datos que serán recibidos en las páginas *JSP* en cuestión, como se determinará a continuación. El objetivo es almacenar esa información en la base de datos, por lo tanto antes de

mostrar la finalización de la comunicación entre los *JSPs* mediante el uso de *jQuery* y *Ajax*, se debe determinar cómo se realiza la conexión con la base de datos.

5.1.3 CONEXIÓN CON LA BASE DE DATOS



```
12  String width = request.getParameter("width");
13  String height = request.getParameter("height");
14  String connectionURL = "jdbc:mysql://localhost:3306/tfg";
15
16  Connection connection = null;
17
18  PreparedStatement pstatement = null;
19
20  Class.forName("com.mysql.jdbc.Driver").newInstance();
21      int updateQuery = 0;
22
23      if(width!=null){
24          if(width!="") {
25                      try {
26                          System.out.println("Conexion");
27          connection = DriverManager.getConnection("jdbc:mysql://localhost/tfg","root", "1234");
28
29          String queryString = "INSERT INTO scenePosition(idScenary, width, height, position) VALUES (?,?,?,?)"
30          pstatement = connection.prepareStatement(queryString);
31          pstatement.setString(1, id);
32          pstatement.setString(2, width);
33          pstatement.setString(3, height);
34          pstatement.setString(4, pos);
35          updateQuery = pstatement.executeUpdate();
36                      if (updateQuery != 0) {  %>
37              <br/>
38              <table style="background-color: #E3E4FA;"WIDTH="30%" border="1">
39                 <tr><th>Data is inserted successfully
40                  in database.</th></tr>
41
42              </table>
43                      System.out.println("Conexion: INSERTADO");
44
45          <%
46
47          }
48        }
49        catch (Exception ex) {
50        System.out.println("Unable to connect to batabase");
51
52          }
53        finally {
54            // close all the connections.
55            pstatement.close();
56            connection.close();
57        }
58      }
59    }
60  %>
```

Ilustración 53. Fragmento de código de conexión con la base de datos mediante JDBC

Se observa como se va a hacer uso de una conexión *JDBC*, tal como se determinó en el capítulo del estado del arte. Para ello hay que utilizar los drivers necesarios para el correcto funcionamiento.

En primer lugar, como se observa en la línea 20, se crea una nueva instancia para la conexión *JDBC*. Después es necesaria la *url JDBC* que contiene la información necesaria para realizar la conexión.

Se crea el objeto *connection*, creando la conexión aplicando la url anteriormente mencionada, se puede observar esto en la línea 27.

```
Connection connection = DriverManager.getConnection(ulr,usuario,clave);
```

INSERCIONES Y MODIFICACIONES

A continuación, una vez hecha la conexión con la base de datos, podemos realizar inserciones, consultas o modificaciones, gracias al uso de *prepareStatement*() (soportado en este caso tanto por el *driver* o conector, como por la base de datos), añadiendo de esta forma un aspecto positivo en cuanto a la seguridad, puesto que con el uso de este método, se realiza automáticamente cualquier conversión teniendo en cuenta comillas, lo que en caso contrario podría dar errores de ejecución en la sentencia SQL y supondría un posible peligro en cuanto a intentos de inyección SQL.

Esto queda determinado desde la línea 29 a la 34, mostradas en la ilustración _.

Por último, se ejecuta el método *executeUpdate*(), actualizando las filas correspondientes en la base de datos.

Sólo queda cerrar la conexión con el fin de evitar el desaprovechamiento de recursos, tal como se muestra en las líneas 53-56.

CONSULTAS

En el caso de las consultas, el sistema es prácticamente similar en cuanto a la creación de la conexión. La diferencia está en que en esta ocasión lo que se realiza es la recuperación de datos de la base de datos, en lugar de la actualización de la misma. Se presenta un ejemplo en la siguiente ilustración.

```
48      <%try
49      {
50
51      Class.forName("com.mysql.jdbc.Driver").newInstance();
52      }
53      catch (Exception ex) {
54      // handle the error
55      System.out.println(" Salta excepcion ");
56      }
57
58      try{
59
60      PreparedStatement st2 = null;
61
62      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost/tfg", "root", "1234");
63
64      st = conexion.createStatement();
65
66      /**** Si es Ok: avisamos ****/
67
68      System.out.println( "Si he llegado hasta aqui es que se ha producido la conexión");
69
70      System.out.println( "Si no se hubiera producido, se habria disparado SQLException");
71
72
73      //IDS
74      ResultSet rs = st.executeQuery("SELECT max(idscenes) from scenes" );
75      id=toArray(rs,"max(idscenes)");
76
77      ResultSet rs2 = st.executeQuery("Select width, height, idScenary from scenes where idscenes =" +id[0
78
79      width=toArray(rs2,"width");
80      height=toArray(rs2,"height");
81      idScenary=toArray(rs2,"idScenary");
82
83      ResultSet rs3 = st.executeQuery("Select position, width, height from scenePosition where idScenary ='
84      position = toArray(rs3, "position");
85      wsc = toArray(rs3, "width");
86      hsc = toArray(rs3, "height");
87      }
88
89      catch (SQLException ex) {
90      // handle any errors
91          System.out.println(" Salta SQL Exception");
92      } %>
93
```

Ilustración 54. Fragmento de código de peticiones a base de datos

Se observa como se realiza la petición, mediante una consulta determinada como en los casos reflejados en las líneas 74, 77 y 83, consultas a tres tablas de la base de datos diferentes. Se utiliza el método anteriormente descrito en la ilustración 50, para el almacenamiento de todos esos valores en *arrays*, con el fin de trabajar con dicha información de forma más cómoda.

5.1.4 GENERACIÓN DEL FICHERO XML

Toda la información recopilada por la aplicación tras la finalización del proceso de creación del escenario debe ser utilizada para la generación del documento XML que será utilizado por otra herramienta externa con el fin de generar el escenario mediante el motor Unity 3D.

Esto se lleva a cabo tal como se indica en la ilustración 55.

```
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2      pageEncoding="ISO-8859-1"%>
3  <%@ page
4      import="java.net.*,java.io.*,java.sql.DriverManager,java.sql.Connection,java.sq
5  <%
6      /*
7       Clase que genera el archivo xml
8       */
9
10      String infoCeldas = request.getParameter("infoCeldas");
11      String[] arrayCeldas = infoCeldas.split("#");
12
13     String celdaUnitaria = "";
14     String filename = "pruebaArchivo.xml";
15     String dataToWrite = "";
16     boolean append = false;
17     try {
18     System.out.println("Creamos el archivo con todos los datos de celda");
                                    (...)
26         FileWriter fw = new FileWriter(filename, append);
                                    (...)
81         /**
82          * Se cierra el archivo
83          */
84         fw.close();
85
86     } catch (IOException ioe) {
87         /**
88          * Se ha producido un error durante la lectura/escritura del archivo
89          */
90         System.out
91                 .println("Se ha producido un error durante la lectura del archivo '
92                         + filename);
93     }
94
95 %>
```

Ilustración 55. Fragmento de código de generación de archivo XML

En la anterior ilustración se puede observar como, mediante la misma comunicación explicada anteriormente basada en *jQuery* y *Ajax*, y el sistema de acceso a la base de datos, se recupera la información desde esta página *JSP*, encargada de generar el documento *XML*.

Queda determinado en la línea 26 el método encargado para la generación del fichero, sobre el que se incluirá el contenido *parseado*, generando etiquetas *XML* y formando el documento que después será utilizado para la representación tridimensional del escenario.

Para realizar el archivo XML en java se utiliza la librería JDOM [38], que da soporte al tratamiento de XML, tanto parseo, como búsquedas, modificación, generación y serialización.

En la figura 56 se muestra un ejemplo del documento XML generado.

```xml
<scene>
<!-- Valores de configuracion globales -->
<tile scene = "1"/>
<floors>
    <rect pos = "1;1;0" size="1;1;1" texture = "blue"/>
    <rect pos = "1;2;3" size="1;1;1" texture = "red"/>
    <rect pos = "1;3;0" size="1;1;1" texture = "blue"/>
    <rect pos = "2;1;3" size="1;1;1" texture = "red"/>
    <rect pos = "2;2;0" size="1;1;1" texture = "orange"/>
    <rect pos = "2;3;3" size="1;1;1" texture = "red"/>
    <rect pos = "3;1;0" size="1;1;1" texture = "orange"/>
    <rect pos = "3;2;0" size="1;1;1" texture = "orange"/>
    <rect pos = "3;3;0" size="1;1;1" texture = "orange"/>
    <rect pos = "4;1;0" size="1;1;1" texture = "orange"/>
    <rect pos = "4;2;3" size="1;1;1" texture = "red"/>
    <rect pos = "4;3;0" size="1;1;1" texture = "orange"/>
    <rect pos = "5;1;0" size="1;1;1" texture = "orange"/>
    <rect pos = "5;2;0" size="1;1;1" texture = "orange"/>
    <rect pos = "5;3;0" size="1;1;1" texture = "orange"/>
    <rect pos = "6;1;3" size="1;1;1" texture = "red"/>
    <rect pos = "6;2;0" size="1;1;1" texture = "orange"/>
    <rect pos = "6;3;3" size="1;1;1" texture = "red"/>
    <rect pos = "7;1;0" size="1;1;1" texture = "blue"/>
    <rect pos = "7;2;3" size="1;1;1" texture = "red"/>
    <rect pos = "7;3;0" size="1;1;1" texture = "blue"/>
</floors>

<entities>
    <spawnpoint>
        <pos>3;0;0</pos>
    </spawnpoint>
    <chest>
        <!-- Definicion de las propiedades de la entidad-->
        <pos>3;0;0</pos>
        <size>1;1;1</size>
        <model>dragon</model>
    </chest>
    <spawnpoint>
        <pos>3;0;0</pos>
    </spawnpoint>
    <chest>
        <!-- Definicion de las propiedades de la entidad-->
        <pos>3;0;0</pos>
        <size>1;1;1</size>
        <model>dragon</model>
    </chest>
    <spawnpoint>
        <pos>3;0;0</pos>
    </spawnpoint>
    <chest>
    <spawnpoint>
        <pos>3;0;0</pos>
    </spawnpoint>
    <chest>
        <!-- Definicion de las propiedades de la entidad-->
        <pos>3;0;0</pos>
        <size>1;1;1</size>
        <model>dragon</model>
    </chest>
    <spawnpoint>
        <pos>3;0;0</pos>
    </spawnpoint>
    <chest>
        <!-- Definicion de las propiedades de la entidad-->
        <pos>3;0;0</pos>
        <size>1;1;1</size>
        <model>dragon</model>
    </chest>
</entities>
</scene>
```

Ilustración 56. Documento XML generado por la aplicación

5.2 RESULTADO FINAL DE LA APLICACIÓN

A continuación se detallará el resultado final de la aplicación, ilustrando cada apartado de dicha herramienta, así como las funcionalidades y operaciones que permite.

En primer lugar se presenta la página inicial, donde se informa al usuario de las acciones principales que puede realizar, permitiendo escoger entre iniciar el proceso de creación del escenario, salir de la aplicación, acceder a un módulo de información sobre la aplicación o acceder al módulo de contacto con el equipo de desarrollo.

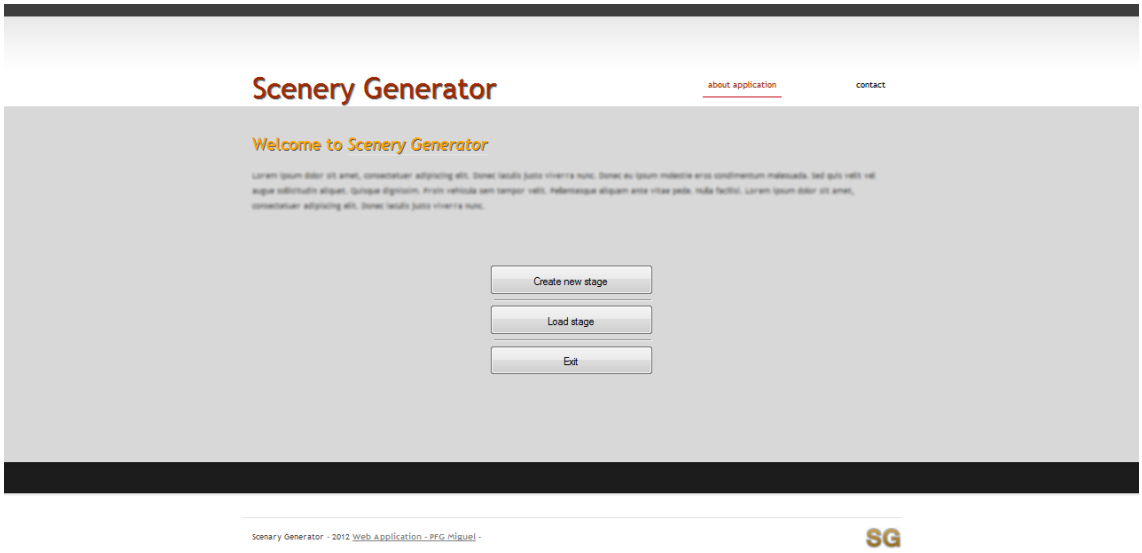


Ilustración 57. Pantalla de inicio de la aplicación

Una vez seleccionada la opción de inicio del diseño de una nueva escena que formará parte del escenario, el usuario verá la siguiente pantalla.

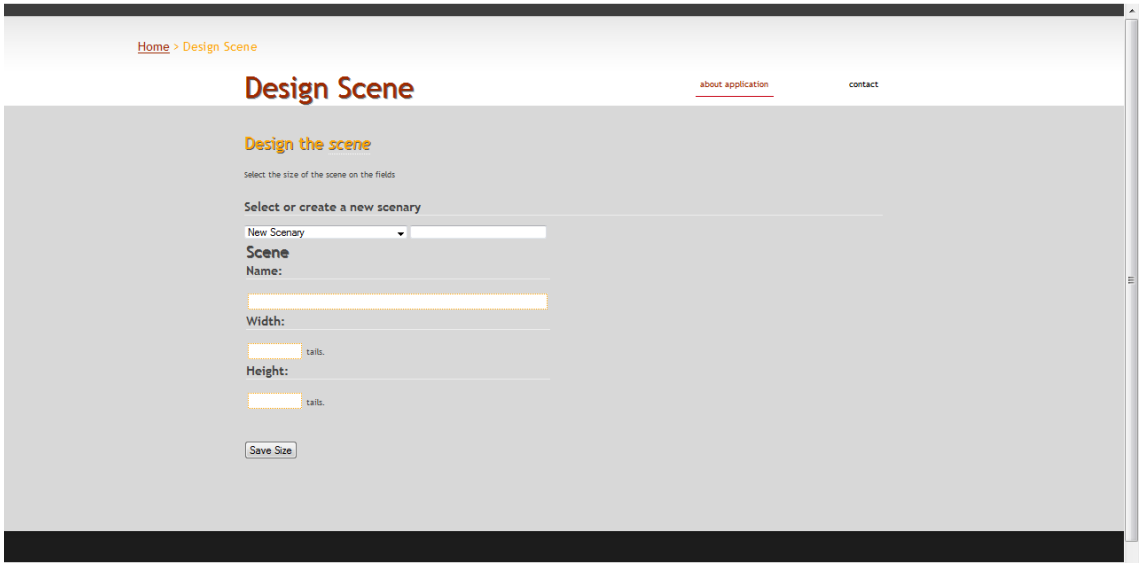


Ilustración 58. Pantalla de diseño de nueva escena

En esta zona de la aplicación, ilustración _, al usuario se le permite la opción de crear un nuevo escenario desde cero, escogiendo en el desplegable la opción denominada como 'New Scenary', permitiendo al usuario, a su vez, la introducción de un nombre identificativo para el escenario (ilustración 59).

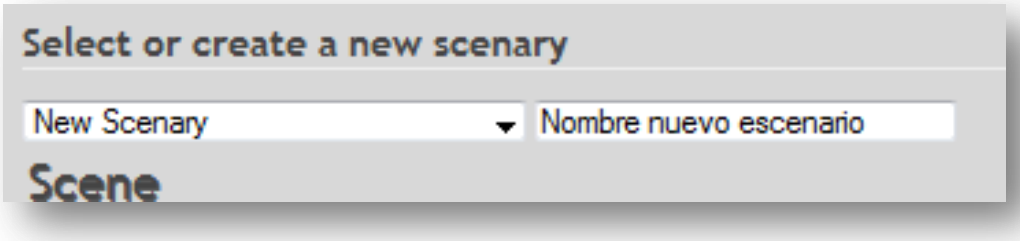


Ilustración 59. Detalle de selección de escenario existente o creación de uno nuevo

A continuación, en la misma pantalla se da opción al usuario de introducir nombre, alto y ancho de la escena que posteriormente pasará a editarse. (Ilustración 60).

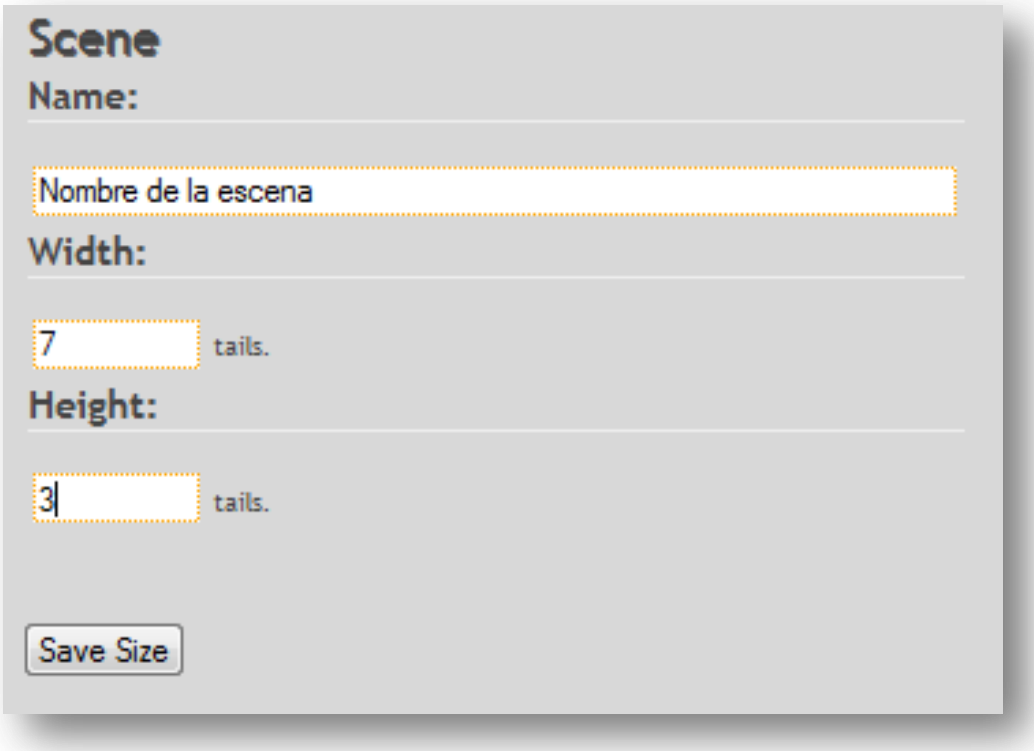


Ilustración 60. Detalle de selección de nombre de la nueva escena y tamaño

Tras un mensaje de confirmación del tamaño de la escena, se procede a mostrar la pantalla de configuración de la misma (ilustración 61), en ella se muestran

dos áreas claramente diferenciadas, la zona de representación gráfica de la escena, mediante una vista cenital; y el área de configuración de texturas, alturas y entidades de la escena (ilustración 62).

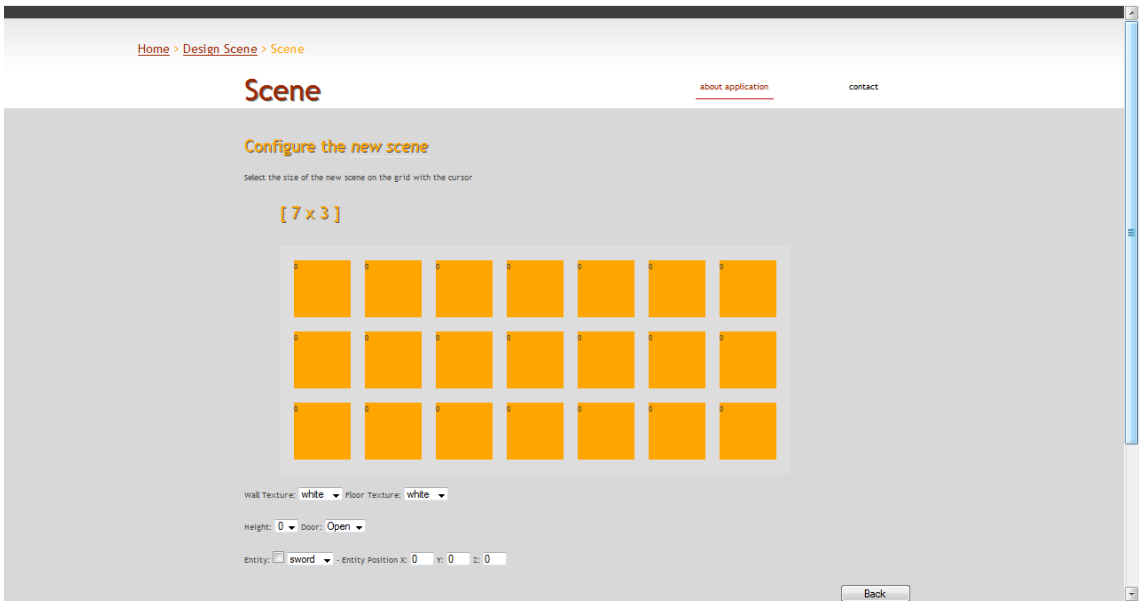


Ilustración 61. Pantalla de configuración de la nueva escena

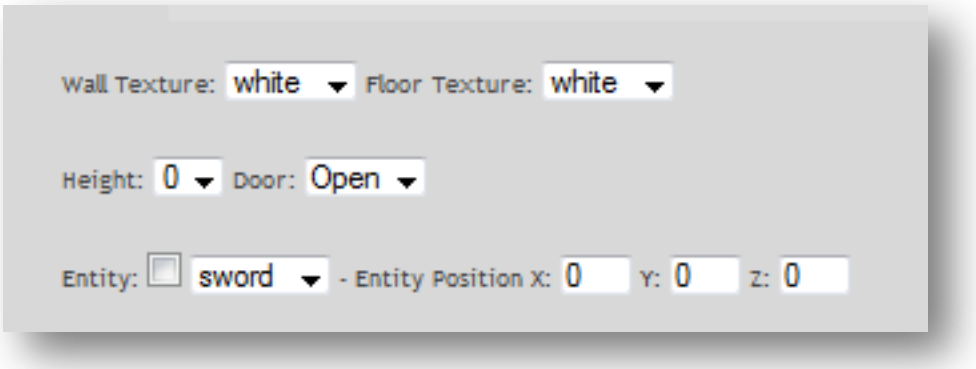


Ilustración 62. Detalle de opciones de configuración de escena

Esta área presenta todas las opciones disponibles para el usuario y la configuración de la escena: elección de texturas de suelos y paredes, altura de los mismos, posición de entidades y determinación de ellas.

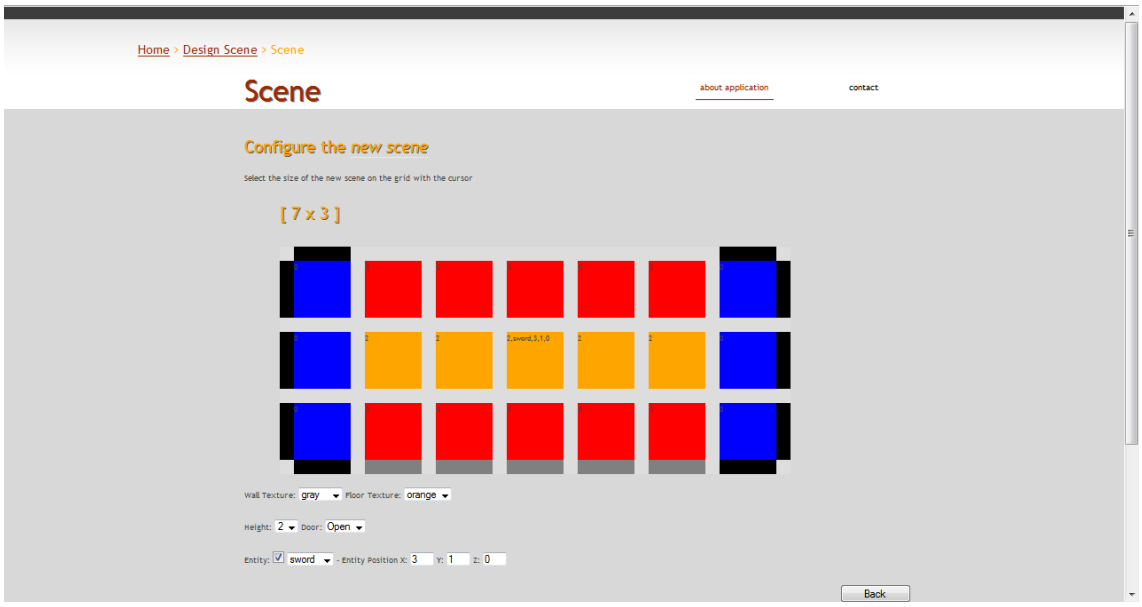


Ilustración 63. Pantalla de configuración de escena II

La siguiente fase corresponde a la situación de la escena recién configurada en el escenario, si se trata de un escenario existente se puede observar en él las escenas previamente creadas.

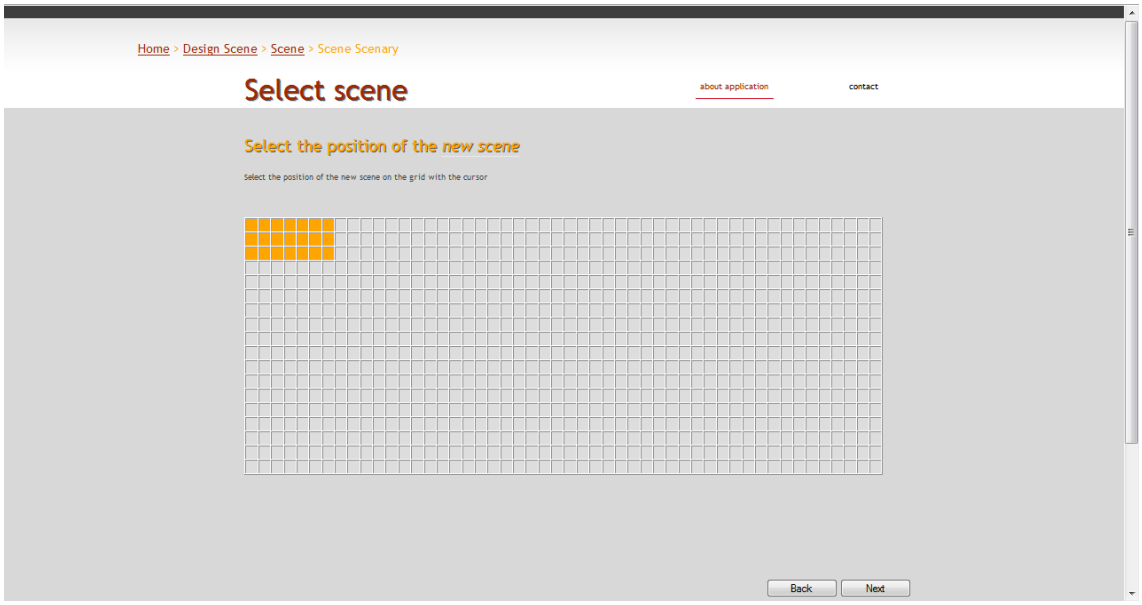


Ilustración 64. Pantalla de situación de escena en el escenario

Una vez conforme la posición de la escena en el escenario, el usuario puede continuar con la edición del mismo, repitiendo el proceso con nuevas escenas.

6. GESTIÓN DEL PROYECTO

En este apartado se procederá a explicar la metodología de desarrollo de software utilizada para la gestión de este proyecto, así como el ciclo de vida seguido, identificando cada fase del desarrollo y el conjunto de actividades planificadas que garantizarán el cumplimiento de objetivos en el periodo de tiempo conveniente. Además, dichas actividades tendrán asociados unos costes determinados y unos recursos que compondrán el presupuesto final del proyecto.

6.1 CICLO DE VIDA DEL PROYECTO

Para la elaboración de este proyecto, se ha seguido un modelo de ciclo de vida en cascada, que identifica el conjunto de fases que comprenden el desarrollo del proyecto en su totalidad, desde el inicio, con el estudio del planteamiento y el desarrollo de la aplicación hasta la fase de pruebas y la documentación redactada sobre la misma.

A continuación se muestran las fases determinadas que componen todo el proceso:

- **Definición y análisis del problema**: Estudio inicial del problema planteado, necesidades a solventar, estudio de plataformas y herramientas a utilizar y planteamiento de objetivos y límites temporales.

- **Análisis del sistema**: Definición de casos de uso y requisitos del software.

- **Diseño del sistema**: Definición de la arquitectura del sistema y componentes que lo forman.

- **Implementación del sistema**: Fase de codificación y desarrollo del sistema, adecuándolo a lo definido en las fases anteriores.

- **Validación del sistema**: Fase de realización de diversas pruebas que verifiquen la integridad, usabilidad y eficiencia del sistema con el objetivo de eliminar posibles errores.

- **Documentación**: Redacción de la memoria del proyecto.

- **Presentación**: Diseño de la presentación del proyecto y preparación de la lectura del mismo.

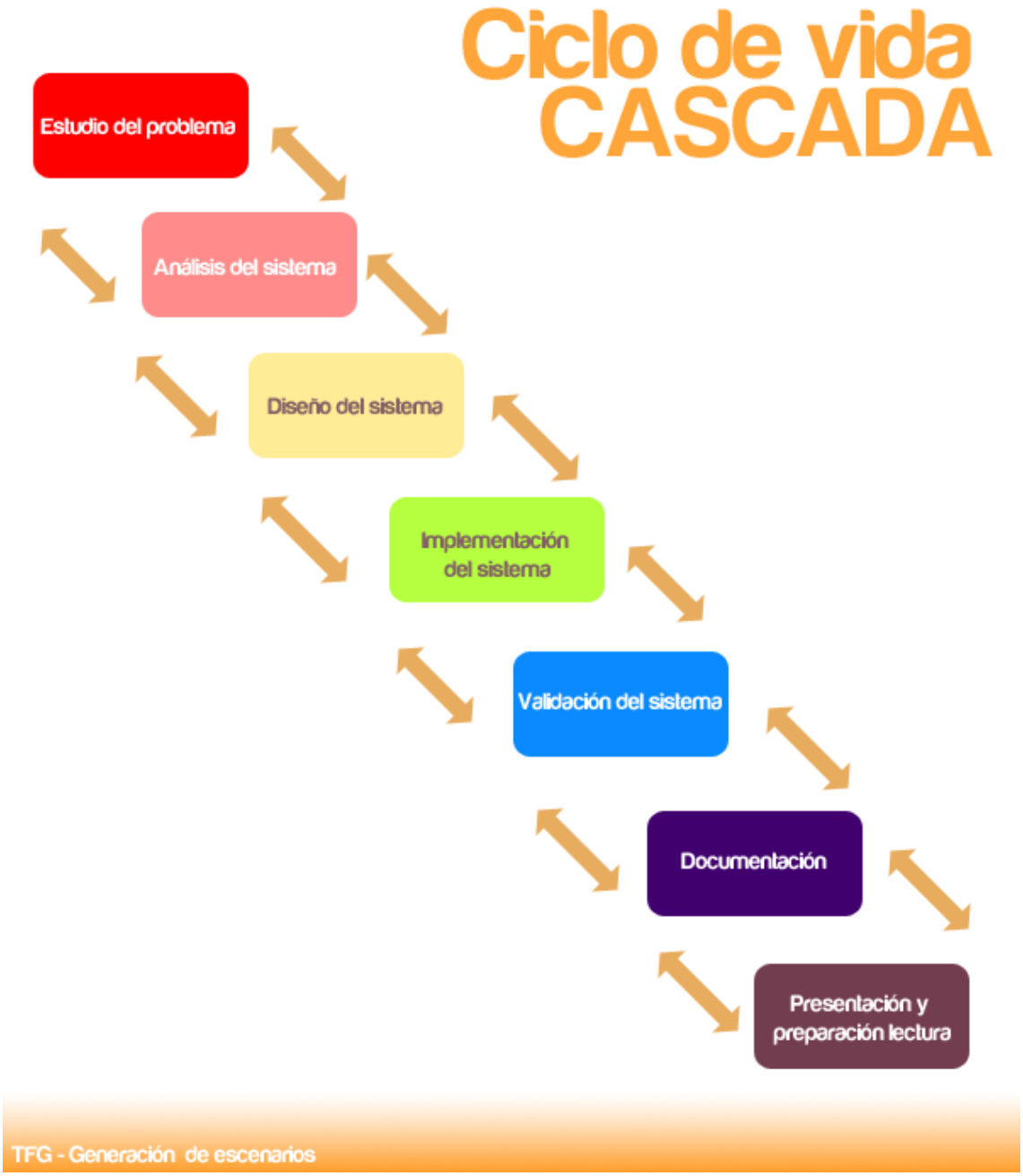


Ilustración 65. Esquema de ciclo de vida en cascada

Tal como se muestra en la ilustración anterior (Ilustración 65), las fases son realizadas secuencialmente, además posee un carácter de retroalimentación, puesto que si se realiza una modificación en alguna de las fases se puede regresar a las anteriores para llevar a cabo los cambios pertinentes, y poder continuar con las siguientes fases. Esto es denominado como ciclo de vida en cascada con retroalimentación.

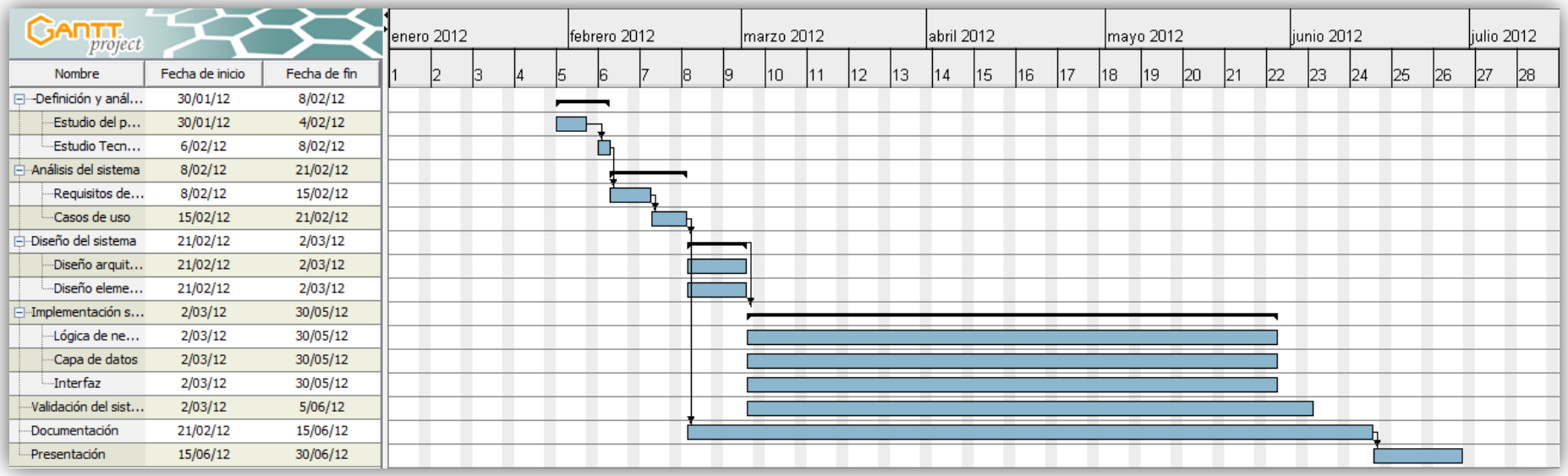
Una vez elegido el modelo de ciclo de vida del proyecto se procedió a planificar cada una de sus fases. El objetivo es la realización de una estimación del tiempo que va a ser utilizado para la realización completa del proyecto.

6.2 PLANIFICACIÓN

En primer lugar se describe la planificación inicial, realizada al principio del proyecto, durante la fase del estudio del problema. La finalidad es la estimación del tiempo necesario para la realización de proyecto y poder analizar la variación una vez finalizado el proyecto entre los tiempos realmente dedicados y la estimación inicial.

6.2.1 PLANIFICACIÓN INICIAL

Esta es la planificación realizada en el mes de febrero de 2012, al inicio del proceso de desarrollo del proyecto.



Nombre	Fecha de inicio	Fecha de fin
Definición y anál...	30/01/12	8/02/12
Estudio del p...	30/01/12	4/02/12
Estudio Tecn...	6/02/12	8/02/12
Análisis del sistema	8/02/12	21/02/12
Requisitos de...	8/02/12	15/02/12
Casos de uso	15/02/12	21/02/12
Diseño del sistema	21/02/12	2/03/12
Diseño arquit...	21/02/12	2/03/12
Diseño eleme...	21/02/12	2/03/12
Implementación s...	2/03/12	30/05/12
Lógica de ne...	2/03/12	30/05/12
Capa de datos	2/03/12	30/05/12
Interfaz	2/03/12	30/05/12
Validación del sist...	2/03/12	5/06/12
Documentación	21/02/12	15/06/12
Presentación	15/06/12	30/06/12

Ilustración 66. Diagrama de Gantt de planificación inicial

En el anterior diagrama de Gantt (Ilustración 66) se puede observar la duración de cada uno de los plazos de realización de tareas, separados por las fases anteriormente descritas. Se puede observar que algunas de las tareas se solapan puesto que pueden ser desarrolladas de forma simultanea, siendo independientes unas con otras, mientras que otras necesitan que las anteriores hayan sido finalizadas para poder iniciarse.

La estimación realizada plantea una duración de casi 5 meses con una duración de 4-5 horas diarias, aproximadamente.

6.2.2 PLANIFICACIÓN FINAL

A continuación se muestran los periodos de tiempo llevados a cabo durante la elaboración del proyecto:

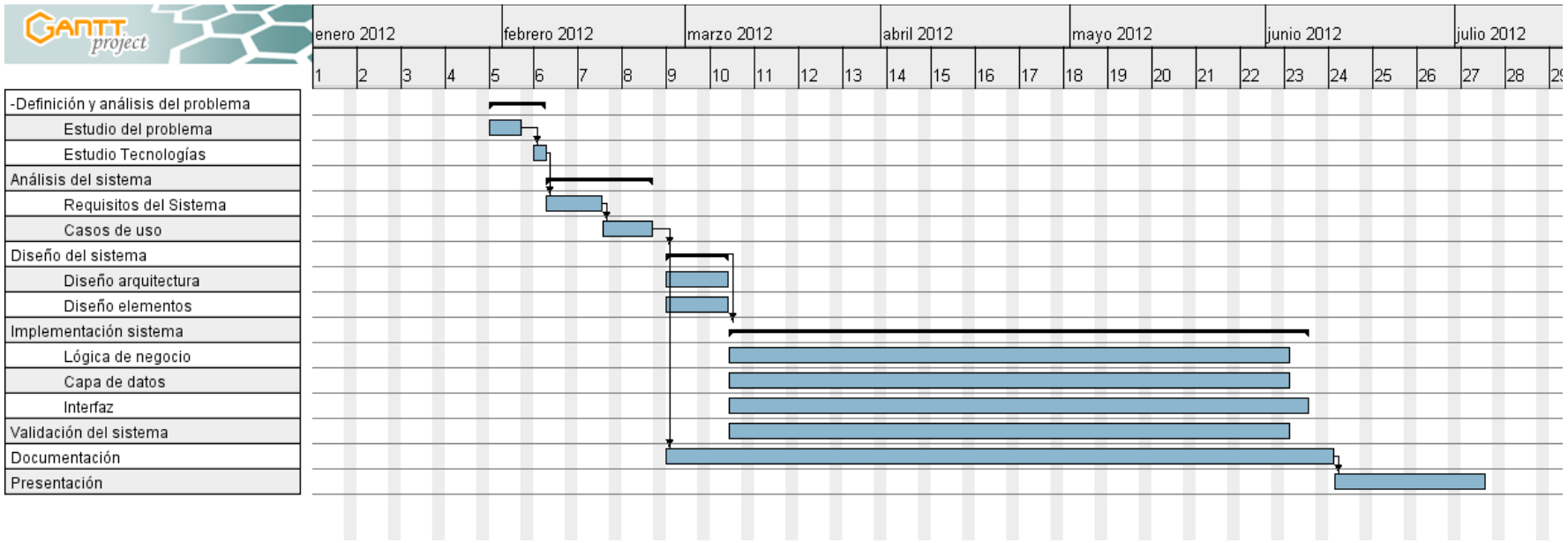


Ilustración 67. Diagrama de Gantt de planificación final

Se puede observar como es bastante semejante a la planificación inicial. Los cambios más destacables se encuentran en el periodo de tiempo de la fase de la implementación, puesto que finalmente fue alargada, teniendo que dedicar más horas a su realización. Al igual que la fase de presentación, puesto que la fecha de lectura del proyecto no se sabía de forma inicial, y se calculó una anterior a la real.

Cabe destacar que la planificación real es más extensa que la inicial, puesto que hubo que alargar el periodo de varias tareas, dedicando más horas a cada una con el fin de poder entregar correctamente el proyecto según los plazos estipulados.

6.2.3 PRESUPUESTO

En esta sección se van a detallar los costes derivados de la realización de las tareas detalladas anteriormente por parte del personal, junto con los surgidos por material necesario para la realización de las mismas.

COSTE DE PERSONAL ENCARGADO DEL PROYECTO

En este caso los encargados del proyecto son dos personas, roles asumidos por parte del tutor y el alumno, donde se puede diferenciar el papel de cliente para el primero y el de diseñador, analista, programador y jefe de proyecto, dependiendo de la fase del mismo, para el segundo.

En la planificación del apartado anterior, se obtenían un total de 159 días, comenzando el 30 de enero de 2012 y finalizando el 6 de Julio del mismo año. Teniendo en cuenta la jornada de trabajo invirtiendo una media de 4 horas, se obtiene un total de 636 horas aproximadamente de dedicación por parte del alumno. Cabe señalar que algunos días, sobre todo los coincidentes con entregas de prácticas finales o preparación de exámenes, no se podía realizar las horas estimadas diarias, pero era solventado en días donde se superaban en gran medida dichas horas, cuando el alumno podía dedicarle más tiempo al proyecto.

Todo esto queda mostrado en el resumen de costes totales de personal, presentado a continuación.

Costes totales de personal			
Rol	**Horas**	**Coste por hora**	**Coste total**
Jefe de proyecto	56	45	2.520 €
Diseñador	60	40	2.400 €
Analista	100	35	3.500 €
Programador	420	20	8.400 €
Total			16.820 €

Tabla 44. Tabla de costes totales de personal

En total, el gasto obtenido derivado de costes de personal al completar el proyecto es de 16.820 euros.

COSTE DE MATERIAL UTILIZADO EN EL PROYECTO

A continuación se muestra el coste asociado al material utilizado para la realización del proyecto.

Costes totales de material				
Material	**Precio de unidad**	**Periodo de amortización**	**Duración del proyecto**	**Coste total**
Equipo Intel Core 2 Duo E660 2.4GHz	420	3 años	4,5 meses	31,12€
Microsoft Windows 7	205	3 años	4,5 meses	15,19€
Microsoft Office 2007	90	3 años	4,5 meses	6,7€
Impresora HP C3180 Photosmart	160	3 años	4,5 meses	11,9€
Total				64,91€

Tabla 45. Tabla de costes totales de material

En total, el gasto obtenido por parte del material asciende a 64,91 euros.

RESUMEN DE COSTES

El coste total de realización del proyecto viene determinado por el siguiente apartado, donde se muestra el coste total derivado teniendo en cuenta tanto el personal como el material utilizado a lo largo del proyecto.

Coste total	
	Coste total
Personal	16.820 €
Material	64,91€
TOTAL	16884,91 €

Tabla 46. Tabla de costes totales

7. CONCLUSIONES Y LÍNEAS FUTURAS

Por último, concluye el documento con una síntesis acerca del desarrollo de la aplicación y las conclusiones obtenidas a lo largo del mismo, añadiendo una propuesta de líneas futuras de mejora.

7.1 CONCLUSIÓN

El rápido proceso evolutivo que sufre en estos momentos nuestra sociedad, provoca que la tecnología esté cada vez más presente en aspectos en los que hace poco ni siquiera se hubiera considerado. Un ejemplo de ello es la educación: en los últimos años hemos asistido a cómo las nuevas tecnologías se han ido abriendo paso en el campo educativo, complementando cada vez más frecuentemente a las técnicas y métodos de aprendizaje tradicionales.

El personal docente cada vez está más concienciado acerca de la importancia de utilizar estas nuevas tecnologías en la educación, y esta es una de las razones que provocan la necesidad de este proyecto. El principal objetivo de este trabajo es proporcionar una herramienta de autoría de creación de escenarios que puedan ser aplicados a videojuegos educativos, de intuitiva apariencia y fácil uso.

El desarrollo de este trabajo se ha llevado a cabo mediante tecnología Java, empleando el entorno de desarrollo Eclipse, una combinación que ha sido analizada en diferentes fases a lo largo del proyecto, y ha resultado ser muy adecuada para la elaboración del mismo.

Cabe destacar como el correcto análisis del sistema, elaborando de forma satisfactoria los requisitos y casos de uso teniendo en cuenta las necesidades presentadas por el tutor en su rol de cliente, ha facilitado el desarrollo de la herramienta, logrando alcanzar los objetivos prefijados al inicio del proyecto, y consiguiendo una herramienta de fácil uso capaz de satisfacer el problema impuesto de forma inicial. Por la misma razón se realizó un elaborado diseño del sistema, sin dejar pasar ningún aspecto relevante, definiendo de forma detallada tanto la arquitectura del sistema y sus módulos, como el prototipo a bajo nivel de la aplicación, ofreciendo una definición visual de la meta que se quería alcanzar.

Una vez finalizadas todas estas fases del proceso, se tiene todas las bases necesarias para la implementación del sistema.

Mediante la elaboración del proyecto, ha surgido el interés por las posibilidades del *game based learning*, encontrando que puede ser muy positivo en el futuro la utilización de estos métodos basados en la educación asistida por ordenador.

Pudiéndose convertir en el complemento perfecto al aprendizaje formal recibido por parte de los alumnos.

En cuanto al aspecto personal, la realización de este proyecto ha servido para profundizar en muchos conocimientos y lecciones conseguidas a lo largo de la carrera, logrando despejar muchas dudas que pudieran haber surgido durante estos años y logrando alcanzar un mayor nivel en cuanto al conocimiento adquirido.

En ocasiones, también han surgido varios problemas tales como conflictos surgidos al inicio del proyecto y relacionados con la configuración del equipo donde iba a ser desarrollado el mismo que, a pesar de haberse llevado con una parte importante del tiempo de dedicación al proyecto, finalmente han podido ser solventados sin mayores consecuencias. Esto hace que esta experiencia sea aún más enriquecedora, y permite sentirse muy realizado.

7.2 LÍNEAS FUTURAS

A pesar de que la aplicación, una vez finalizado su desarrollo, cumple con los requisitos establecidos al inicio del proyecto, es cierto que pueden añadirse múltiples funcionalidades que pueden convertirla en una herramienta de autoría más completa en algunos aspectos.

Una mejora a aplicar en un futuro sería la de perfeccionar en la visualización de la composición de la escena, haciéndola todavía más intuitiva mediante el apoyo de más elementos gráficos, sustituyendo la representación simbólica de entidades por iconos más representativos, que acerquen dicha visualización al resultado real una vez generado el juego.

También puede ser interesante aplicar un módulo con carácter de red social, en el caso de que la aplicación sea utilizada por más de un educador, en que cada uno pueda puntuar y opinar sobre las demás creaciones de los usuarios restantes, formando una comunidad en la que cada uno pueda enriquecerse de las opiniones ajenas con el fin de mejorar el resultado.

8. REFERENCIAS Y BIBLIOGRAFÍA

[1] - D. Druckman, "The educational effectiveness of interactive games", D. Crookall & K. Arai (Eds.), Simulation and gaming across disciplines and cultures: ISAGA at a watershed (pp. 178-187). Thousand Oaks, CA: Sage. 1995

[2] - S. Chen and D. Michael, "Games for Physical and Mental Health." - Serious Games: Games That Educate, Train, and Inform. Boston, MA: Thomson Course Technology; (2005). http://www.gamasutra.com/features/20051031/chen_01.shtml

[3] - P. Moreno-Ger, D. Burgos, J. L., Sierra, B. Fernández-Manjón: "Educational Game Design for Online Education". Computers in Human Behavior 24, 2530–2540 (2008)

[4] - M. Macedonia, 2002. "Games soldiers play", IEEE Spectrum. 30(3): 32–37

[5] - "Seeking Reusability of Computer Games Designs for Informal Learning" - Telmo Zarraonandia, Paloma Díaz, Ignacio Aedo, Mario Rafael Ruíz - Computer Science Department, Universidad Carlos III de Madrid

[6] - "Videojuegos y educación" - Félix Etxeberria Balerdi - Universidad del País Vasco
http://campus.usal.es

[7] - Constatado en las investigaciones llevadas a cabo en la Universidad de Valencia, dirigidas por P.M. Perez Alonso-Geta. En el año 1992, se analizaron 1.600 niños de toda España, con edades comprendidas entre 9 y 14 años.

[8] – "Videojuegos y educación, revisión crítica de la investigación" – A. Mendiz, J. Pinedo, J. Ruiz, J. Mª. Pulido
http://ares.cnice.mec.es/

[9] – "Telegames teach more than you think" – G. H. Ball, 1978

[10] – Constatado en un experimento realizado por J. Griffith en 1983 entre un grupo de estudiantes de enseñanza primaria.

[11] - Greenfield,-Patricia-M; Cocking,-Rodney-R (Eds.) (1996): *Interacting with video. Advances in applied developmental psychology,* Stamford , (U.S.A.): Ablex Publishing Corp. & University of California , Dept. of Psychology, Los Angeles (California)

[12] - Casey J.: "Counseling Using Technology with At-Risk Youth". *ERIC Digest* . Office of Educational Research and Improvement (ED), Washington , DC . 1992
http://ares.cnice.mec.es/

[13] – Brain Training (Nintendo) – Dr. Kawashima.
http://www.nintendo.es

[14] – English Training (Nintendo) – Dr. Kawashima.
http://www.nintendo.es

[15] – Torque 3D - http://www.garagegames.com/products/torque-3d

[16] – Unity 3D - http://unity3d.com/

[17] – Motores gráficos para videojuegos – Jordi Catà, 2002. Departament d'Informàtica i Matemàtica Aplicada – Universitat de Girona
http://www.udg.edu/depima

[18] – Luis Fernández Sanz – Universidad Europea de Madrid – Asociación Técnica de Informáticos– Tecnología, ingeniería del software. " El futuro de la tecnología del software".
http://www.ati.es/

[19] - Construyendo aplicaciones web con una metodología de diseño orientada a objeto - Darío Andrés Silva, Bárbara Mercerat. [LIFIA] – Laboratorio de Investigación y Formación en Informática Avanzada. Facultad de Informática. UNLP.
http://www.lifia.info.unlp.edu.ar/es/

[20] – Aplicaciones Web y de Escritorio, ¿Diferencias en el desarrollo? – Sergio Tarrillo, Universidad Nacional de Trujillo - http://geeks.ms/blogs/sergiotarrillo

[21] – Java http://www.java.com/es/

[22]-The Java Language Specification, Third Edition http://java.sun.com/docs/books/jls/

[23] – TIOBE - http://www.tiobe.com/

[24] - J2EE http://java.sun.com/javaee/

[25] – Eclipse - http://www.eclipse.org/

[26] – Netbeans - http://netbeans.org/

[27] - .NET – Microsoft http://www.microsoft.com/net

[28] – Java & C# Comparison - Harding University - http://www.harding.edu/

[29] – Apache Tomcat - http://tomcat.apache.org/

[30] – GlassFish Server - http://glassfish.java.net/es/

[31] - OpenGIS Consortium (1999) OpenGIS Simple Features Specification For SQL
http://www.opengis.org/docs/99-049.pdf

[32] – Sistemas de Gestión de BBDD y SIG – Universidad de Murcia - http://www.um.es/

[33] - MySQL - http://www.mysql.com/

[34] – JDBC Drivers - http://devapp.sun.com/product/jdbc/drivers

[35] – Sistemas Cliente – Servidor – Informática – Universidad de Vigo, Área de Ciencias de la Computación e Inteligencia Artificial - http://ccia.ei.uvigo.es/

[36] – Arquitecturas Cliente – Servidor. – Universidad de Alicante, Dpto. Sistemas Informáticos http://www.dlsi.ua.es/

[37] – jQuery - http://jquery.com/

[38] – JDOM - http://www.jdom.org/